



P r o f e s s i o n a l E x p e r t i s e D i s t i l l e d

Oracle WebLogic Server 12c: First Look

A sneak peak at Oracle's recently launched WebLogic 12c,
guiding you through new features and techniques

Michel Schildmeijer

[PACKT] enterprise 
PUBLISHING professional expertise distilled

Oracle WebLogic Server 12c: First Look

A sneak peek at Oracle's newly launched WebLogic 12c, guiding you through new features and techniques

Michel Schildmeijer



BIRMINGHAM - MUMBAI

Oracle WebLogic Server 12c: First Look

Copyright © 2012 Packt Publishing

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior written permission of the publisher, except in the case of brief quotations embedded in critical articles or reviews.

Every effort has been made in the preparation of this book to ensure the accuracy of the information presented. However, the information contained in this book is sold without warranty, either express or implied. Neither the author, nor Packt Publishing, and its dealers and distributors will be held liable for any damages caused or alleged to be caused directly or indirectly by this book.

Packt Publishing has endeavored to provide trademark information about all of the companies and products mentioned in this book by the appropriate use of capitals. However, Packt Publishing cannot guarantee the accuracy of this information.

First published: June 2012

Production Reference: 1150612

Published by Packt Publishing Ltd.
Livery Place
35 Livery Street
Birmingham B3 2PB, UK.

ISBN 978-1-84968-718-8

www.packtpub.com

Cover Image by Asher Wishkerman (a.wishkerman@mpic.de)

Credits

Author

Michel Schildmeijer

Proofreader

Stephen Swaney

Reviewers

Vivek Acharya

Wickes Potgieter

Indexer

Monica Ajmera Mehta

Acquisition Editor

Rukshana Khambatta

Graphics

Valentina D'Silva

Manu Joseph

Lead Technical Editor

Unnati Shah

Production Coordinator

Aparna Bhagat

Technical Editor

Manasi Poonthottam

Cover Work

Aparna Bhagat

Project Coordinator

Joel Goveya

About the Author

Michel Schildmeijer, was born in the Netherlands, in the hot summer of 1966. He has lived his entire life in the capital, Amsterdam. After mid-school, he started studying pharmacy. After four years, he had to fulfill his military duty, at the Royal Dutch Air force, working in a pharmacy.

After this period, he got a job as a Quality Inspector at a Pharmacy Company, but after about two years, he switched his job for a position in a hospital's Pharmacy, where he worked for over 10 years.

In the meantime, he got married to Tamara and got two boys, Marciano and Robin. His personal life wasn't always that easy, because his wife got extremely ill for some period, so he had to take all responsibilities for managing his family. Fortunately, he got intensive support from his parents-in-law, who helped greatly in taking care of his kids.

During his Pharmacy job, around 1994, he got acquainted with the Medical Information System which was taking care of structuring patient medical history and information. This was a system running on HP UNIX, a MUMPS SQL database and text-based terminals. He started learning UNIX and MUMPS to give operational support. By then he became enthusiastic, so he switched jobs and started working for some IT companies. Around 2000, he started using Oracle on a big banking application for settlements and clearance. The system was running on Oracle 7 and AIX UNIX and BEA WebLogic and BEA Tuxedo. This was the first time he worked with WebLogic. From then on, he got more and more specialized in Middleware and Oracle. He worked on many projects. Around 2006, he started working on several projects for IBM, in the Oracle Middleware team, administering, configuring, and tweaking large Oracle Middleware systems with Oracle SOA Suite, Oracle Portal, Oracle HTTP, and many more.

In 2008 he began working for Randstad Holding, and got more and more specialized in developing the middleware infrastructure around applications. He started an investigation about migrating the Oracle Application Server 10g and SOA Suite 10g to the 11g platform. Around that period, Oracle acquired BEA.

From working in Brussels for Belgacom, a big Telco company in Belgium, he started his current job, Oracle Fusion Middleware Architect, for AMIS, an IT Company specialized in Oracle and Java.

His focus was always at developing the infrastructure for many companies, advising them how to migrate or build a new middleware platform based on the latest 11g techniques. He also became an instructor, teaching all the basics of Oracle WebLogic.

The reason for him to write this book is to get familiar with the new features in WebLogic 12c, and because he thinks it's a great product with a lot of new features, especially the new Java EE 6 features and Exalogic optimizations.

Michel is now working for Qualogy as a member of the Exalogic Squad Team.

Qualogy is an international organization delivering both standard and custom Oracle and Java solutions and services.

Qualogy uses first-rate applications and works with solid partners and highly-qualified consultants who are more than willing to offer their know-how to further improve your organization. This results in customized automation that ensures the business processes within your organization will run more efficiently and simpler than ever before.

Qualogy offers optimum support during the whole automation process: from advice, development, and testing to implementation and monitoring.

He specializes in Oracle, Java/JEE, Consultancy, Oracle eBusiness Suite, Exalogic, Web2.0, and QAFE.

I would like to thank some people who helped me in completing this book:

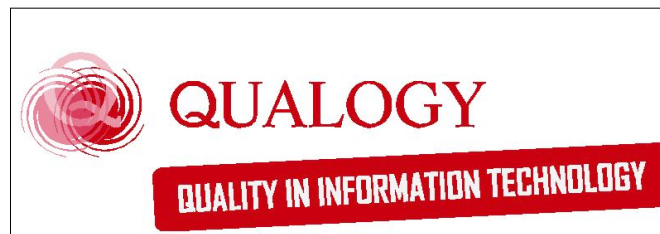
My wife Tamara, whose life is a difficult struggle sometimes

Janny and Steef, who took care of my kids

Marciano and Robin, my great kids

All the reviewers

And those who supported me in an unusual way



Michel Schildmeijer is an Oracle Fusion Middleware Architect at Qualogy.

Oracle Platinum Partner Qualogy has in-depth expertise in delivering Oracle-based technologies and services, including advanced technologies such as Oracle Fusion Applications, Oracle Fusion Middleware, and Oracle Exalogic Elastic Cloud. Qualogy was founded in The Netherlands in 1998. Today the company is home to over 150 specialists in the field of Java and Oracle Development, Oracle E-Business Suite, Fusion Middleware, Oracle Exalogic, Database Administration, Business Intelligence, Agile Consultancy, SOA, Big Data, Cloud and Web development with Enterprise Application Platform QAFE (<http://www.qafe.com>). We provide tailor-made applications and a wealth of expertise for integrating, streamlining, and providing insight into complex business processes.

Qualogy is ISO 9001 certified, showing customers that processes have been documented in a system of quality, and that the company can quickly track, correct, and prevent possible errors.

Additionally, Qualogy is NEN 4400 certified, Certified Oracle Solution Partner, W3C member, Google Apps Authorized Reseller, and Top ICT Employer for a couple of years in a row.

For more information, please visit www.qualogy.com.

About the Reviewers

Vivek Acharya is an Oracle Consultant working as a professional freelancer. He has been in the design, development, consulting, and architect world for approximately seven years working in Oracle Practice at GE, IBM, and HP. He is an Oracle Certified Expert as Oracle Fusion-SOA 11g Implementation Specialist and Oracle - BPM 11g Implementation Specialist. He has experience and expertise in Oracle Fusion – SOA, BPM, BAM, Mediator, B2B, BI, AIA, Web logic, workflow, Rules, WebCenter, ECM, IDM, Oracle fusion applications, SaaS, On Demand, and so on. He loves all things to do with Oracle Fusion Applications, Oracle SOA, Oracle BPM, Cloud Computing, Sales force, SaaS, and BSM.

He has been the author of a couple of books on distributed systems, Oracle BPM, and so on, and keeps an interest in playing synthesizer and loves travelling. You can add him at <http://www.linkedin.com/pub/vivek-acharya/15/377/26a>, write to him at vivek.oraclesoa@gmail.com, and read him at <http://acharyavivek.wordpress.com/>.

Wickes Potgieter has worked as a product specialist for over 12 years. His main focus was on the BEA WebLogic suite of products and after the Oracle acquisition of BEA Systems, he focused on the Oracle Fusion Middleware suite of products. His experience ranges from solution architecture, infrastructure design, administration, development, pre-sales, and training to performance tuning of the Oracle Fusion Middleware products, JVM, and custom applications. He specializes in Oracle WebLogic Server, JRockit, Service Bus, SOA, BPM, BAM, Enterprise Manager 11g/12c, WebCenter, Identity and Access Management, and Application Performance Management.

They have formed a specialized consulting company in 2003 with offices in the United Kingdom and South Africa, covering customers in the EMEA region. They are an Oracle Gold partner and have a team of specialized Oracle Fusion Middleware consultants servicing customers both onsite and offsite.

TSI-Systems website: www.tsisystems.co.uk and Wickes can be contacted at wickes@tsisystems.co.uk.

I would like to thank my wife, Mary Jane, for her patience and assisting me through all the late nights. Thank you to all my friends and family for constant encouragement.

www.PacktPub.com

Support files, eBooks, discount offers and more

You might want to visit www.PacktPub.com for support files and downloads related to your book.

Did you know that Packt offers eBook versions of every book published, with PDF and ePub files available? You can upgrade to the eBook version at www.PacktPub.com and as a print book customer, you are entitled to a discount on the eBook copy. Get in touch with us at service@packtpub.com for more details.

At www.PacktPub.com, you can also read a collection of free technical articles, sign up for a range of free newsletters and receive exclusive discounts and offers on Packt books and eBooks.



<http://PacktLib.PacktPub.com>

Do you need instant solutions to your IT questions? PacktLib is Packt's online digital book library. Here, you can access, read and search across Packt's entire library of books.

Why Subscribe?

- Fully searchable across every book published by Packt
- Copy and paste, print and bookmark content
- On demand and accessible via web browser

Free Access for Packt account holders

If you have an account with Packt at www.PacktPub.com, you can use this to access PacktLib today and view nine entirely free books. Simply use your login credentials for immediate access.

Instant Updates on New Packt Books

Get notified! Find out when new books are published by following [@PacktEnterprise](#) on Twitter, or the *Packt Enterprise* Facebook page.

Table of Contents

Preface	1
Chapter 1: Ready for the Cloud!	7
The c is replacing the g	7
WebLogic 12c supports over more than 200 new features!	8
Overview and structure in the new features	8
Java EE 6 support and development	8
Java EE 6 features	8
Development features	9
Configuration and tooling	11
Performance and failover	11
Traffic management	12
Enterprise Manager 12c	12
Distributed caching	13
Some more Exalogic features	14
Summary	15
Chapter 2: Supporting the Java EE 6	17
Java EE 6 applications for conventional and cloud deployment	17
Major Java EE 6 API changes	18
Java EE 6 specifications	19
Contexts and Dependency Injection for Java EE (JSR 299)	19
Java Server Faces (JSF) 2	21
Enterprise Java Beans 3.1	22
Admin console support for EJBs in a WAR	26
EJB 3.1 annotation support	26
Simplified deployment with annotation support	26
Bean Validation 1.0 (JSR 303)	28
Java Persistence API (JPA) 2	28
Servlets 3.0	31
Java API for RESTful Web Services (JSR 311)	32

Java EE Connector Architecture 1.6	33
Deprecated APIs	33
WebLogic 12c shared libraries and modules	34
Java classes compatibility	35
Summary	36
Chapter 3: Deployment, Installation, and Configuration Features	37
Develop, build, compile, and deploy on WebLogic 12c	37
Lightweight development with WebLogic 12c	38
Some hints and tips using development on WebLogic 12c	39
Using FastSwap	39
Using the wlx option	40
Using WebLogic server tooling	40
Standard Java IDE support	41
Eclipse and Oracle Enterprise Pack for Eclipse (12.1.1.0)	41
NetBeans IDE 7.1	43
Other expected IDEs	44
WebLogic 12c and Maven integration	44
The project object model (POM)	45
Advanced features of WebLogic Maven plugin	48
Maven support for several IDEs	49
Maven for Eclipse/OEPE	49
NetBeans and Maven	50
Classloading and the Classloading Analysis Tool (CAT)	51
Overview of Java EE application Classpath	51
Built-in WLS CAT (ClassLoading Analysis Tool)	52
Deployment descriptor support for GlassFish Server	54
Cloud development with WebLogic 12c	55
Installation and upgrades with WebLogic 12c	55
Upgrading to WebLogic 12c	57
New configuration features in WebLogic 12c	58
JDK 7 certification	58
Administration Console	58
NodeManager	58
JDBC	58
Security	59
Standalone clients	60
Deprecated: weblogic.management.username and weblogic.management.password	60
Web Services	60
Exalogic features	61
WebLogic 12c New feature TLog Store	62
Summary	62

Chapter 4: Integrated and External Services	63
JDBC services	63
Active GridLink and RAC integration	63
Fan enabling	64
New JDBC features for WebLogic 12c	65
JMS Services	69
Security services	70
Java Authentication Service Provider Interface for Containers (JASPIC) support	70
RSA JSSE Provider	72
SSL Implementation	72
Changes to SSLMBean	73
JSSE/SSL	73
TLS 1.2 support	73
Better support for Single Sign-On with Microsoft Clients	74
Web Services	75
WebLogic Web Services with Java EE 6	75
WebLogic 12c and Jersey JAX-RS RI	
Version 1.9	76
Support for EclipseLink MOXy (JAXB)	77
Summary	78
Chapter 5: Integration and Management with Enterprise Manager 12c Cloud Control	79
What is Oracle Enterprise Manager 12c?	79
Oracle Enterprise Manager 12c system design	80
WebLogic Server Management: New in Enterprise Manager 12c	81
Configuration management features	82
WebLogic Server 12c provisioning and cloning	82
Automating discovery and detecting configuration changes	85
WebLogic Server 12c monitoring	86
Performance monitoring and diagnostics of WebLogic Server	86
Customizable performance summaries	88
Out-of-box metrics	88
Metric Extensions	89
Composite Application dashboard	90
Request Monitoring	90
JVM Diagnostics	91
Middleware Diagnostics Advisor	93
Diagnostic Snapshots	94
Monitoring for deployed applications	95

Table of Contents

Application components dependency and performance	96
Log Viewer	97
Event monitoring	98
Business Transaction Management	98
Heap Analysis	99
Integrated Cloud Stack Management	99
Summary	99
Chapter 6: Oracle WebLogic 12c to the Cloud: Exalogic	101
What is Oracle Exalogic?	102
Exabus	104
Oracle Exalogic Cloud Software components	104
Exalogic Cloud Software	105
RDMA API: Oracle Tuxedo	106
Message Bus API: Oracle Coherence	108
SDP API: WebLogic	109
Oracle Virtual Assembly Builder	110
Oracle Traffic Director	111
Oracle WebLogic/Exalogic optimizations	114
Increased server scalability, throughput, and responsiveness	114
Better Oracle RAC and Exadata integration	115
Reduced Exalogic to Exadata response times	116
Summary	117
Index	119

Preface

Oracle WebLogic 12c is Oracle's number one strategic Application Server—able to run on both cloud computing systems and conventional ones. Oracle WebLogic 12c implements the new Java EE 6 standard and supports Java SE 7, and this book will guide you through all the new features, enhancements, and tools inside the new 12c release.

Oracle WebLogic Server 12c: First Look offers a focused look at the new Weblogic features with real-world examples.

This practical guide gives clear explanations and dives deep into all the definitions and concepts of WebLogic 12c.

This book starts with a short introduction to WebLogic 12c. It then swiftly covers the new features of Java EE and SE where we will also learn to develop Java EE 6 applications. This book also covers the new configuration and deployment features. Finally, all the new cloud features and techniques will be highlighted, including integration with Enterprise Manager 12c.

What this book covers

Chapter 1, Ready for the Cloud!, gives you an overall introduction to the new WebLogic 12c and its new features. All the topics discussed later in this book will be introduced in here so you will know what to expect later on.

Chapter 2, Supporting the Java EE 6, covers some of the new features of Java EE 6 and SE and which features are used in WebLogic 12c and how they fit in into this new 12c release.

Chapter 3, Deployment, Installation, and Configuration Features, discusses other major or minor improvements that will appear, like different types of installations, domain configurations, new deployment plugins, and strategies like the Maven plugin, and also explains Oracle Virtual Assembly Builder.

Chapter 4, Integrated and External Services, covers new integrated services such as Coherence, JDBC, JMS, and all kinds of new or enhanced security services in 12c. It also discusses Active GridLink for JDBC, Partitioned Distributed Destinations for JMS, and many others which will give you a good overview of all kinds of new, enhanced, or deprecated services.

Chapter 5, Integration and Management with Enterprise Manager 12c Cloud Control, discusses the role of Enterprise Manager 12c Cloud Control and what it can deliver for Middleware Administrators about monitoring and configuring your WebLogic Server environment.

Chapter 6, Oracle WebLogic 12c to the Cloud: Exalogic, discusses the role of WebLogic Server 12c in Oracle's Engineered system, Exalogic, and topics about the hardware and software components in an Exalogic box.

What you need for this book

The following is the list of what you need for this book:

- WebLogic Server 12c(12.1.1) for Linux or Windows, or the generic JAR version
- A JDK like JRockit or HotSpot
- Oracle Enterprise Eclipse Server pack 11g
- NetBeans 7.1.1
- Enterprise Manager 12c

Who this book is for

If you are a WebLogic Server administrator or developer excited about the new features introduced in the 12c version, then this is the guide for you. A working knowledge of previous WebLogic versions is preferable.

Conventions

In this book, you will find a number of styles of text that distinguish between different kinds of information. Here are some examples of these styles, and an explanation of their meaning.

Code words in text are shown as follows: "boot.properties should be created manually when running in the Production mode and should be placed in the Domain directory in the security folder of the Admin Server".

A block of code is set as follows:

```
<path id="wlappc.classpath">
    <fileset dir="${wl.home}/server/lib">
        <include name="*.jar"/>
    </fileset>
</path>
```

When we wish to draw your attention to a particular part of a code block, the relevant lines or items are set in bold:

```
<fileset dir="${wl.home}/server/lib">
    <include name="*.jar"/>
</fileset>
```

Any command-line input or output is written as follows:

```
java weblogic.appc -verbose -keepgenerated .
[JspcInvoker]Checking web app for compliance.
[jspc] Overriding descriptor option 'keepgenerated' with value specified
on command-line 'true'
[jspc] -webapp specified, searching . for JSPs
[jspc] Compiling /index.jsp
```

New terms and **important words** are shown in bold. Words that you see on the screen, in menus or dialog boxes for example, appear in the text like this: "There's a new section here, **Transaction Log Store**".



Warnings or important notes appear in a box like this.



Tips and tricks appear like this.

Reader feedback

Feedback from our readers is always welcome. Let us know what you think about this book – what you liked or may have disliked. Reader feedback is important for us to develop titles that you really get the most out of.

To send us general feedback, simply send an e-mail to feedback@packtpub.com, and mention the book title via the subject of your message.

If there is a book that you need and would like to see us publish, please send us a note in the **SUGGEST A TITLE** form on www.packtpub.com or e-mail suggest@packtpub.com.

If there is a topic that you have expertise in and you are interested in either writing or contributing to a book, see our author guide on www.packtpub.com/authors.

Customer support

Now that you are the proud owner of a Packt book, we have a number of things to help you to get the most from your purchase.

Downloading the example code

You can download the example code files for all Packt books you have purchased from your account at <http://www.PacktPub.com>. If you purchased this book elsewhere, you can visit <http://www.PacktPub.com/support> and register to have the files e-mailed directly to you.

Errata

Although we have taken every care to ensure the accuracy of our content, mistakes do happen. If you find a mistake in one of our books – maybe a mistake in the text or the code – we would be grateful if you would report this to us. By doing so, you can save other readers from frustration and help us improve subsequent versions of this book. If you find any errata, please report them by visiting <http://www.packtpub.com/support>, selecting your book, clicking on the **errata submission form** link, and entering the details of your errata. Once your errata are verified, your submission will be accepted and the errata will be uploaded on our website, or added to any list of existing errata, under the Errata section of that title. Any existing errata can be viewed by selecting your title from <http://www.packtpub.com/support>.

Piracy

Piracy of copyright material on the Internet is an ongoing problem across all media. At Packt, we take the protection of our copyright and licenses very seriously. If you come across any illegal copies of our works, in any form, on the Internet, please provide us with the location address or website name immediately so that we can pursue a remedy.

Please contact us at copyright@packtpub.com with a link to the suspected pirated material.

We appreciate your help in protecting our authors, and our ability to bring you valuable content.

Questions

You can contact us at questions@packtpub.com if you are having a problem with any aspect of the book, and we will do our best to address it.

1

Ready for the Cloud!

Anyone who follows the Middleware world, and especially the application server market, should have noticed the change on December 1, 2011.

This date was chosen by Oracle to announce the launch of its next generation of Fusion Middleware products, using 12 as the major release number.

One of the first products to be released for the version-12 family along with the launch of the Enterprise Manager 12c – the Java Enterprise Application Server – forms the foundation of Oracle's Fusion Middleware product Oracle WebLogic Server 12c!

Oracle WebLogic Server is already known as Oracle's strategic number one application server for JAVA Enterprise Applications, and is the first which will be at the 12c release. Later on in 2012, other products from the Oracle family, such as the Oracle SOA Suite, will follow.

The *c* is replacing the *g*

As you can see, Oracle replaced the *g* in the release with *c*. It all had to do with where Oracle put their focus. The *g* stood for grid computing which Oracle introduced starting from release 10. Oracle's grid computing product group includes (among other things) a database management system (DBMS) and an application server. In addition to supporting grid computing features such as resource sharing and automatic load balancing, 10g products automate many database management tasks. The Real Application Cluster (RAC) component makes it possible to install a database over multiple servers. Oracle has done a lot of effort to get ready for cloud computing, the *c* appears in the main release.

Oracle also aligned their internal release numbers, where as in 11g it was a bit confusing, for example, Oracle WebLogic 11g R1 PS 4 stood for version 10.3.5, now internal release-number is 12.1.1

WebLogic 12c supports over more than 200 new features!

Those who had followed the launch of the new Oracle WebLogic 12c on December 1, 2011, should have seen all commercial and marketing one-liners that they've launched in the diverse presentations, demos, and webcasts. One of them was:

Of course, discussing all those 200 new features would make this book a 1000 pages thick, but the headlines will be handled in this book and we will zoom in on some really important features.

Overview and structure in the new features

To bring some structure in all the new features, we will divide them in categories from which you will get a clearer view, and address the new features in a broader perspective.

Java EE 6 support and development

The new Oracle WebLogic 12c implements the Java EE 6 standards, which supports all kinds of Java EE 6 specifications implemented such as Java EE 6 features and development features:

Java EE 6 features

The new features and specifications are listed as follows:

- JSF 2.0
- Java Servlets 3.0
- JPA 2.0
- EJB 3.1
- JAX-RS
- Managed Beans 1.0
- Support for Java SE 7 (and Java SE 6) which includes:
 - Java language optimizations and internationalization
 - Client and server support
 - SSL/TLS 1.2 in JSSE to support JAVA socket transport security
 - JVM Converge

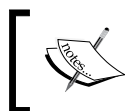
Not typically a specification, but important though: the convergence of the Java VM; JRockit and HotSpot, are both incorporated with the best features from both Java virtual machines. The JVM convergence will be a multiyear process. Probably it will be a converged JVM-based on HotSpot with all goodness of JRockit.

The following are the JRockit features and specifications:

- Robustness
 - Cooperative thread suspension
 - More robust JIT compiler
 - White box testing APIs
 - Refactored codebase for maintainability
- Serviceability
 - JRockit Flight Recorder
 - HPROF heap dump support
 - Enhanced JMX agent
 - Native memory tracking
 - Fine granular compiler directives
- Performance
 - Up to 64 GB compressed references (was 4 GB)
 - Up to 30 percent lower GC pause times overall

The following are the features of HotSpot JVM:

- Oracle apps and middleware on Solaris
- Client and non-Oracle apps on Solaris/Windows/Linux



We will discuss JVM Converge in detail in *Chapter 2, Supporting the JAVA EE 6*. We will also highlight some of the important new JAVA EE 6 specifications.

Development features

WebLogic 12c has support for many IDEs. WebLogic already supports JDeveloper 11.1.1.5, but will come out with 11.1.1.6 later on. However, anyone developing applications using WLS 11.1.1.5 can deploy them to WLS 12c.

Also supported are Eclipse and NetBeans 7.1 IDE. As said, the JDeveloper 11.1.1.6 and IntelliJIdea IDE will be supported in a later timeframe. The following is the screenshot of the IDEs already supported by WebLogic 12c:



The following screenshot shows the IDEs that will be supported later in 2012. Unfortunately, during the writing process of this book, JDeveloper 12 was not available.



The following are the other features:

- New enhanced WebLogic Maven plugin.
- Lightweight development with WebLogic server. The ZIP distribution file does not contain any installers and can be used to configure a domain when unzipped.
- Built-in GlassFish descriptor recognition for easy re-deployment to Oracle WebLogic Server. GlassFish Server supports the `weblogic-application.xml`, `weblogic.xml`, and `weblogic-webservices.xml` deployment descriptor files.

Configuration and tooling

There are many new features and tooling in the new WebLogic 12c. They are as follows:

- WebLogic 12c provides upgrades from iAS with automated tooling WebLogic 11g
- GlassFish redeployment with a built-in GlassFish descriptor recognition for re-deployment to WebLogic server
- JBoss and WebSphere with migration services
- WebLogic 12c has Active GridLink – This is an optimization for RAC Databases. GridLink uses Fast Connection Failover for faster RAC failure detection.

Performance and failover

As WebLogic 12c is part of the Cloud foundation, it will run on conventional systems, and also on utilized hardware or better called engineered systems or Exalogic. To meet the requirements of these new hardware techniques, WebLogic 12c has better performance features.

Some of these features are:

- Higher Performance accomplished with different kinds of techniques such as the following:
 - Parallel muxers with Java NIO APIs for low-level I/O-based operations
 - An optimized work scheduler providing improvements to the Increment Advisor used to manage the size of WebLogic Server's self-tuning thread pool
 - Lazy de-serialization of session data on the replica server until required
 - Multiple replication channels for synchronous in-memory session replication between servers in a WebLogic cluster
 - Enhanced high availability and disaster recovery

Traffic management

WebLogic 12c supports Oracle Traffic Director. Oracle Traffic Director is a layer-7 software load balancer. Oracle Traffic Director:

- Is a load balancer
- Is a local traffic manager
- Uses the application network layer 7
- Can act as a reverse proxy
- Supports SSL 3.0 and TLS 1.0. You can configure SSL/TLS-enabled HTTP listeners
- Will be the replacement for Oracle Web Cache
- Is not a built-in feature of WebLogic 12c
- Supports Integrated traffic management such as routing, load balancing, request-routing and caching, and SSL crypto acceleration

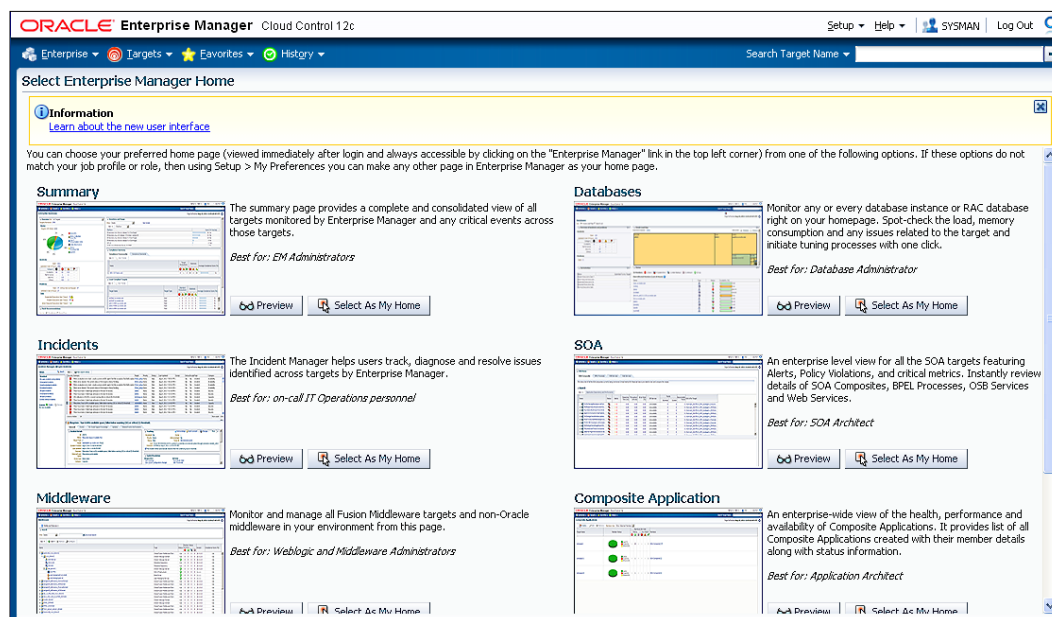
Enterprise Manager 12c

Tight integration with the Enterprise Manager 12 Cloud Control and the use of the Middleware. Within the Enterprise Manager one can administer, clone, perform deployment, and provision tasks. Enterprise Manager 12c will be discussed in *Chapter 5, Integration and Management with Enterprise Manager 12c Cloud Control*.

Some of the other features are as follows:

- Navigate the middleware routing topology
- Customize middleware performance summaries
- End-to-end performance management
- Use the middleware diagnostics advisor to size the JDBC connection pool
- Diagnose WebLogic performance bottlenecks
- Capture diagnostics snapshots
- Clone an Oracle WebLogic domain from the software library
- Deploy a Java EE application
- Manage SOA suite
- Manage Coherence

The following screenshot shows you a typical middleware diagnostics page:



Managing Oracle WebLogic Server with EM 12c provides you a broad end-to-end monitoring and management perspective from the external face of applications, to the majority of the business logic. This means multiple clusters of managed servers that handle both presentation and business logic and communicate with each other via RMI, Web Services, and other remote invocations in order to complete transactions for frontend processes. In order to properly manage these WebLogic servers (whether there are multiple large-scale deployments or just a couple of clusters), administrators need to keep track of performance, service levels, configurations, error/exception handling, patching, and general application life cycle activities such as scale out and WebLogic domain or Java EE application provisioning.

Distributed caching

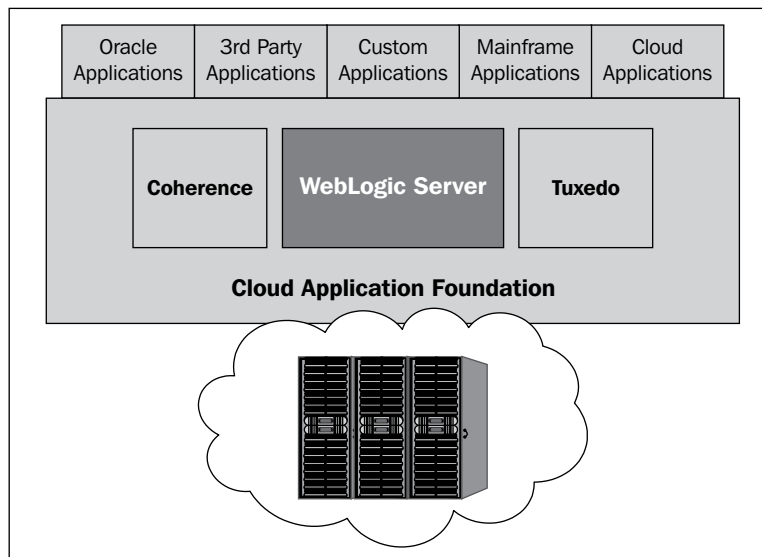
The use of Coherence already exists, but was always a bit of a side product, from WebLogic 12 there will be a tighter integration with the latest Coherence as follows:

- ActiveCache integration for JPA used in WebLogic server. This allows JPA Entity caching and the TopLink grid enables you to direct queries to the Coherence Active Cache.
- Coherence clusters have their own MBeans within WLS, which means more integration of Coherence into the WebLogic server

Another integration is the one with the node manager, which is used for starting/stopping cache servers remotely and from the console. Exalogic and Cloud ready!

Oracle WebLogic is an Enterprise Application Server part of the Application Cloud Foundation. Oracle Cloud Application Foundation combines technologies together: Oracle Exalogic Elastic Cloud, the basis for the cloud world, Oracle WebLogic Server for Java EE, Oracle Tuxedo for C/C++/COBOL, Oracle Coherence in-memory data grid, Oracle JRockit and Hotspot Java SE solutions, Oracle Enterprise Manager, Oracle Virtual Assembly Builder, and Oracle Traffic Director.

In the following diagram, you can see where WebLogic is positioned in this foundation:



Some more Exalogic features

The following are some Exalogic features:

- The Virtual Assembly Builder: Deploys, un-deploys scale assemblies with Oracle Virtual Assembly, quickly create and configure entire multitier application topologies. With OVA, there is a new model for deployment, patching, versioning, and management.

- Exabus: High-speed network virtualization. Exabus has the following components:
 - Coherence 3.7 for Java applications
 - Tuxedo 12c for C++ applications
 - Infiniband network interface

Both are using direct memory access and kernel bypass for better throughput and lower latency.

- Enterprise Manager 12c Cloud Control, which we have seen earlier in this chapter.

More Exalogic features will be discussed in *Chapter 6, Oracle WebLogic 12c to the Cloud – Exalogic*.

Summary

The new Oracle WebLogic 12c has done a massive transformation with some of the most important features such as Java EE 6 and Exalogic readiness. Oracle has made a huge step into the future launching their number one Application Server to the next level, and I think you will agree when I say that this is heaven on earth for an IT technician.

In the next chapters, we will do a deep dive into the various features, with sometimes a side-step to some to WebLogic related products.

2

Supporting the Java EE 6

One of the most exciting new features in the new Oracle WebLogic 12c is that it supports the Java EE 6 specifications.

In this chapter, we will have a look at the new Java EE 6, but more in particular how they fit in into Oracle WebLogic 12c and the Application Cloud Foundation. Also, we will have a look at how this fits in applications working on the Exalogic platform, along with WebLogic 12c.

The main thought or strategy of Oracle for WebLogic 12c is to, as they say, develop modern, lightweight Java EE 6 applications. So let's see if Oracle has accomplished this strategy with the new WebLogic 12c.

Java EE 6 applications for conventional and cloud deployment

As said, Oracle has put in great efforts to make WebLogic 12c cloud-ready. However, that does not mean conventional systems will be left behind. It could be the case that a company decides not to have a cloud, either private or public.

Some goals for the Java EE 6 platform are as follows:

- Being flexible and lightweight: Providing some lightweight interfaces such as JAX-RPC, EJB 2.x Entity Beans, JAXR, JSR 88
- Extensible: To be more open and flexible, it embraces open source frameworks
- Easier to use and develop on: Already this path was set on Java EE 5 and continued in Java EE 6

Major Java EE 6 API changes

Every Java EE 6 API has been changed, enhanced, or some minor updates have taken place. The focus of this version was put on WebTier. For this purpose there is a Web 2.0 profile with some of the interfaces to make it able to use the lightweight features. The use of profiles with specific subsets of Java EE APIs are intended for specific types of applications.

Each profile is fully integrated and just works out of the box, although integrating add-ons is still possible. With profiles one can create modular, lightweight Java EE compliant application servers a lot easier. In this release, there's only one profile, the Web Profile (except from the full profile where you get all the APIs that belong to the full profile.)

The following table shows you which API's are in the Web Profile:

API	Web Profile
Servlet 3	√
JSF 2	√
CDI	√
EJB 3.1	√
JPA 2	√
Bean validation	√
JTA	√

The major changes the new Java EE 6 has are:

- Contexts and Dependency Injection (CDI): Since the introduction of Java EE 6, CDI is the next generation dependency injection.
- Java Server Faces (JSF) 2: JSF 2 is more flexible, is easy to use and has adopted various new technologies and new features.
- Enterprise Java Beans (EJB) 3.1: Ease of use, some new features are added. EJB Lite is a lightweight version with local interfaces, session beans (stateful, stateless and singleton), and no timers or scheduling included.
- Java Persistence API (JPA) 2: More flexibility and new features.
- Servlet 3: Easier to use, new features that are focused on lightweight web profile. Also, some more security enhancements and asynchronous support.
- Java API for RESTful Web Services (JAX-RS 1.1): REST-based web services in addition to SOAP support in JAX-WS.

- Bean validation: Validating data in JavaBeans can be done better by expressing application constraints declaratively.
- Java Connection Architecture 1.6 (JCA): Ease of development by using metadata annotations and no need to use `ra.xml` anymore, better security, and integration for EIS applications.
- Java Authorization Contract for Containers 1.4 (JACC): Some updates like the use of annotations for propagating security policies.
- JAXB 2.2: To bind an XML schema to Java and the other way around was updated too.

In the following sections, some of the new features will be highlighted.

Java EE 6 specifications

As mentioned earlier, Oracle WebLogic 12c supports Java EE 6, which supports a lot of new features. Some of these features are discussed here.

Contexts and Dependency Injection for Java EE (JSR 299)

This specification is a generic dependency injection with automatic context management. It's an integral part of Java EE 6 and provides an architecture that allows Java EE components such as servlets, enterprise beans, and JavaBeans to exist within the lifecycle of an application with well-defined scopes. In addition, CDI services allow Java EE components such as EJB session beans and JSF-managed beans to be injected and to interact by firing and observing events. When you specify this option, it generates `beans.xml` in the `WEB-INF` folder of your application. The `beans.xml` file is used by CDI to instruct the Oracle WebLogic Server that the project is a module containing CDI beans.

When the application is deployed, it knows there is a `beans.xml` file, so the classes on the path are scanned for CDI annotations.

You now can create a `ManagedBean` and add a stateless annotation for a simple EJB.

In the following example you can see how the `inject` class is being used:

```
package demo;

import javax.inject.Named;

@Named
```

```
public class MessageServerBean {  
    public String getMessage() {  
        return "Hello World!";  
    }  
}
```

And with this, you can add `MessageServerBean` in a facelet as follows:

```
<h:body>  
    Hello from Packt  
    <br/>  
    Message is: #{messageServerBean.message}  
    <br/>  
    Message Server Bean is: #{messageServerBean}  
</h:body>
```

Another example being used within the WebLogic Server is the Resource Adapters Bean discovery. This discovery detects if a Resource Archive (RAR) is a bean archive. WebLogic will then treat all JAR files inside the RAR as bean archives, and even overrules the `META-INF/beans.xml` files in this case. The bean archive descriptor indicates that a specific RAR is a bean archive.

Some components that support CDI within RARs are as follows:

- **ResourceAdapter bean**—RAclass that uses the `javax.resource.spi.ResourceAdapter` interface, with operations for life cycle management and message endpoint setup
- **ManagedConnectionFactory bean**—The JavaBean class that uses the `javax.resource.spi.ManagedConnectionFactory` interface and is a factory of both `ManagedConnection` and EIS-specific connection factory instances
- **ActivationSpec bean**—The JavaBean class that uses the `javax.resource.spi.ActivationSpec` interface contains the activation config info for a message endpoint
- **Admin objects**—Set of classes that represents objects specific to a messaging style or message provider

The following steps explain how an injection does its work for a resource adapter like the `DBAdapter`:

1. It initializes the RA component bean configuration properties from deployment descriptors.
2. Then the `PostConstruct` annotation after dependency injection is done to perform any initialization.

3. Performs bean validation and invokes the `validate()` method.
4. For an RA bean, invokes the `start()` method.
5. Makes all resource adapter component beans available either by binding them to JNDI or exposing them to endpoint applications.

Java Server Faces (JSF) 2

As you all Java developers probably know, JavaServer Faces is a web-based framework in the presentation layer that provides a subset of components for graphical user interface, which binds user interface components according to an event-driven model to objects.

JSF is a standard user interface (UI) framework for developing Java EE web applications. It contains a default set of UI components, custom tag libraries for adding UI components to a view, a server-side event model, and managed beans (state management).

JSF 2 is already supported from WebLogic 10.3.3. In a typical WebLogic Server installation, you will find the supported JAR files under the WLS Server Home, in the directory `common/deployable-libraries/jsf-2.0.war`.

The JSF 2.0 shared library follows the same model as the previous JSF libraries shipped with WLS, where it needs to be deployed and referenced using a library ref; in a WLS deployment descriptor by applications that wish to use it.

The JSF 2.0 library supports Dependency Injection of Java EE resources and the use of the Java EE 5 lifecycle annotations in managed beans as described in the specification. This is done through the inclusion of a WLS-specific class that implements the `com.sun.faces.spi.InjectionProvider` interface provided in the `WEB-INF/lib/wls.jsf.di.jar` library within the `jsf-2.0.war` shared library.

Some new features in JSF 2 are:

- Enables JSF views in XML: In this feature, the JSP document file can be treated as facelet file
- Pluggable Facelet Cache mechanism: In JSF 2.1.2, the in-memory cache of the `Facelet` instance is served from a cache that is overridden with an implementation in this API
- System events
- Enhanced navigation
- GET support
- Bean validation

- Proper error handling
- Project stages
- First class JavaScript and Ajax support
- Component behaviors
- Resource handling
- Delta state saving
- Tree visiting
- Annotation-based configuration
- Content Dependency Injection support

Beware, when using JSF 2.1 in your WebLogic Server, it requires at least a Servlet 3.0 container.

The JSF implementation has been added directly to the WebLogic Server classpath. This is a change from the WebLogic Server 11g release. In this change, the JSF implementation was provided as an optional shared library, which needed to be deployed in order for the applications to use JSF. With WebLogic Server 12c, JSF is now an integral part of the server and can be used without the necessity of deploying and referencing the shared library.

Even with the new JSF 2, WebLogic 12c supports the older JSF 1.2 and JSTL 1.1 packages. They are bundled with WebLogic Server as shared libraries, though they are deprecated in this release. Existing web applications that use JSF 1.2 and JSTL 1.1 functionality can run on WebLogic Server. Choose the appropriate JSF or JSTL library based on your application.

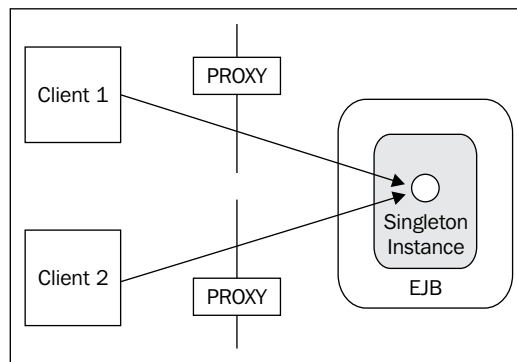
In this release, the `weblogic.xml` file in `jsf-1.2.war` configures a filtering class loader for your application's JSF classes and resources. Do not forget to make a library reference in `weblogic.xml` in your application.

Enterprise Java Beans 3.1

EJBs are managed beans with additional services like transactions. They provide distributable and deployable business services to clients

The purpose of EJB 3.1 along with its JSR 318 specification was to simplify the development and implementation of EJB. This simplification will hit two key areas: development and packaging. This goal matches exactly with the new Java EE 6 because of these features, ease of development, ease of use, and lightweighted.

- Singleton beans with concurrency control:** Singleton sessions have the ability to share application session state between multiple instances of an EJB. A singleton bean bounds a session bean once per application in a particular JVM, for the life cycle of the application. It can be useful to employ a scheme in which a single shared instance is used for all clients. And new to the EJB 3.1 Specification is the singleton session bean to fit this requirement. The following diagram explains how clients can use the singleton bean to share the state of a counter service. A stateless singleton bean can be called from a Java client, with the count being consistently incremented.



- Cron-style declarative and programmatic Timers:** When you have some knowledge of UNIX or Linux, you probably know the meaning of cron. It's a scheduler from which you can schedule an action at any time and as often as you wish.

Already in 3.0 there was a timer scheduler, this is enhanced in 3.1.

The most important one in this set of enhancements is the ability to declaratively create cron-like schedules to trigger EJB methods.

The following is an example of such a scheduler:

```

@Stateless
public class ReleaseDateCounterBean implements ReleaseDateCounter{
    @Schedule(second="0", minute="0", hour="0", dayOfMonth="1",
    month="*", year="*")
    public void generateMonthlyNewsLetter() {
        ... Still .. days to go before this book is published...
    }
}
  
```

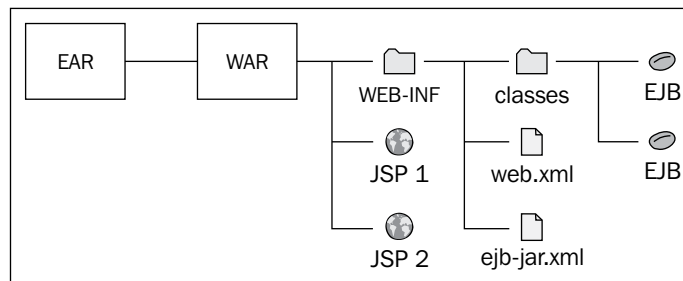
Look at the typical cron-style here:

```

@Schedule(expression="0 0 0 1 * * *")
  
```

- **Simplified WAR packaging:** Now another interesting feature is that you can package your EJB as part of WAR. EJBs can be directly dropped into the `WEB-INF/classes` directory and deployed as part of the WAR. In a similar vein, the `ejb-jar.xml` deployment descriptor, if you happen to be using one, can be placed into the `WEB-INF` directory along with the `web.xml` file. It may also be possible to place an EJB JAR into the `WEB-INF/lib` directory.

The following diagram shows the simplified packaging schedule:



- **Portable global JNDI names:** For standardizing access to EJB applications, portable global JNDI has been introduced.

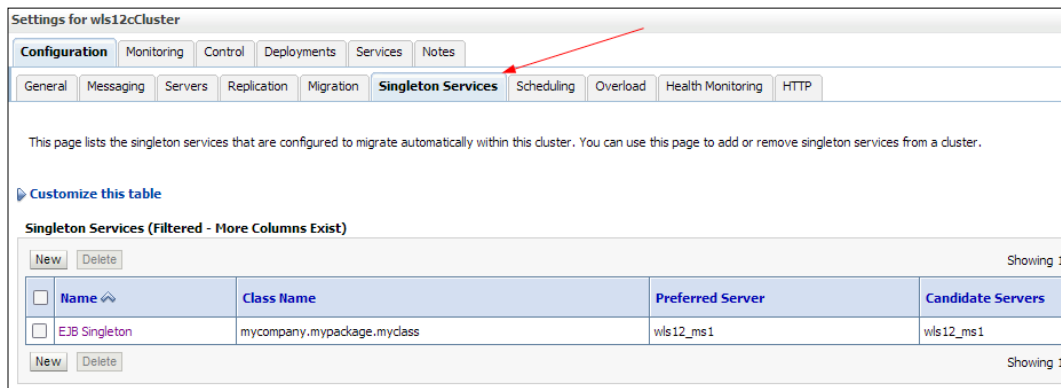
EJB components can be registered and looked up from using the following pattern:

```
java:global[/<app-name>]/<module-name>/<bean-name>
```

This can be important when EJBs are accessed locally but are deployed in a WebLogic cluster. The JNDI can be accessed from every cluster member using the standardized global application's name. A client running in the same Managed Server Instance as a bean instance uses the same API to access the bean as a client running in a different Managed Server Instance on the same or different machine.

- **Startup/shutdown callbacks:** Singleton beans also provide a way for EJBs to receive callbacks during application initialization or shutdown. By default, the container decides when to instantiate the singleton instance. However, you can force the container to instantiate the singleton instance during application initialization by using the `@Startup` annotation. This gives the bean permission to define a `@PostConstruct` method to be called at startup time. At last, any `@PreDestroy` method for a singleton is guaranteed to be called when the application is shutting down.

In a WebLogic cluster, you can create the singleton service. And also specify your application's class.



- **EJB Lite**—a standard lightweight subset of the EJB API: EJB Lite is, in fact, as the name suggests, EJB with some disabled features as much as possible. On one hand, this allows for very simple, lightweight implementations. On the other hand, this means learning EJB Lite could consist of learning just a small handful of annotations and almost no configuration. The next generation of lightweight Java EE application servers will probably implement EJB Lite instead of the entire EJB 3.1 specification.

The following is the current list of features supported for EJB Lite:

- Stateless, stateful, and singleton session beans
- Only local EJB interfaces or no interfaces
- Interceptors
- Declarative security
- Declarative and programmatic transactions
- **Embeddable EJB:** An API for executing EJB components within Java SE.

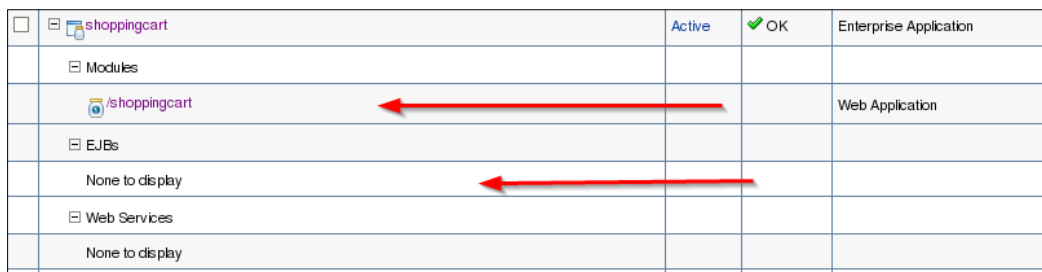
The following options are also included in the EJB 3.1 specification:

- Embedded Containers/Testing support
- Calendar-based timer expressions
- Asynchronous bean invocation

Next we will discuss some of the additional options in WebLogic 12c.

Admin console support for EJBs in a WAR

As you can see in the following screenshot, WAR and EJB are combined into one module instead of a separate one. In this case, it's easier to administer the application because everything is in one place now.



<input type="checkbox"/>	shoppingcart	Active	OK	Enterprise Application
	Modules			
	/shoppingcart			Web Application
	EJBs			
	None to display			
	Web Services			
	None to display			

EJB 3.1 annotation support

The annotated bean is, in fact, the control center of your EJB. It contains the Java code about how your EJB behaves. It is a Java class file that takes care of implementing the business logic and methods of your EJB. You are able to annotate this bean file with metadata from the JDK to make a specification of shape and characteristics of the EJB, and define services as enhanced business-level security or special business logic during runtime.

Simplified deployment with annotation support

EJB 3.1 provides a dramatically simpler deployment with support for deployment inside a WAR file. A class with a component-defining annotation becomes an enterprise bean component when packaged within `WEB-INF/classes` or as `.jar` in `WEB-INF/lib`. Enterprise beans may also be defined using a `WEB-INF/ejb-jar.xml` file. Beans packaged in `.war` share a single namespace, and become part of the WAR's environment. Packaging a JAR in `WEB-INF/lib` is thus semantically equivalent to putting the classes in `WEB-INF/classes`.

To provide a simple compile, build, and deploy, you could make use of the `weblogic.appc` class. This is not a new feature, but I want to mention this tool because you can use it to be able to quickly and easily build and compile your EAR files.

The `appc` tool offers you the flexibility of compiling an entire application, instead of compiling segments. WebLogic Server has access to all modules during the EAR compilation. If an error occurs while running `appc` from the command line, `appc` exits with an error message. By running `appc` prior to deployment, you potentially reduce the number of times a bean is compiled.

After setting the right environment variables on your WebLogic Server, using `setDomainEnv` or `setWLSEnv` you can run the following command:

```
java weblogic.appc [options]
```

Else create an ant task to use it in ant.

```
ant task weblogic.ant.taskdefs.j2ee.Appc

<project basedir="." default="appc" name="appc">
  <property environment="env"/>

  <property name="wl.home" value="/app/wlserver_12.1"/>
  <property name="application.dir" value="/home/weblogic/workspace/
wls12c/testWeb/WebContent" />

  <echo message="${wl.home}/server/lib"/>

  <path id="wlappc.classpath">
    <fileset dir="${wl.home}/server/lib">
      <include name="*.jar"/>
    </fileset>
  </path>

  <echo message="${toString:wlappc.classpath}"/>

  <taskdef name="wlappc" classpathref="wlappc.classpath"
classname="weblogic.ant.taskdefs.j2ee.Appc"/>

  <target name="appc">
    <wlappc source="${application.dir}"
      keepgenerated="true"
      verbose="true"/>
  </target>
</project>
```

Now you can run the appc tool from the command line.

```
java weblogic.appc -verbose -keepgenerated .
[JspsInvoker]Checking web app for compliance.
[jspc] Overriding descriptor option 'keepgenerated' with value specified
on command-line 'true'
[jspc] -webapp specified, searching . for JSPs
[jspc] Compiling /index.jsp
<Jan 29 4, 2012 4:38:03 PM CET> <Info> <J2EE> <BEA-160220> <Compilation
completed successfully>
```

Bean Validation 1.0 (JSR 303)

Working along with other Java specifications such as Context Dependency Injector or Java Persistence API is Bean Validation.

Although it came out under Java EE 6, it does not need Java EE 6 to function.

Bean Validation defines a metadata model and API for the `JavaBean` validation. The metadata source is annotations, with the ability to override and extend the metadata through the use of XML validation descriptors.

Within managed beans in JSF applications, WebLogic Server does support the use of dependency injection. The simplest way to get it to work is to use the documented mechanism of deploying. WebLogic Server supplies the JSF shared library, with the specific implementation of the `com.sun.faces.spi.InjectionProvider`. The `InjectionProvider` implementation is automatically configured on the web container.

The new EE managed beans specification defines a base component model for Java EE, together with a very basic set of container services (`@Resource`, `@PostConstruct`, `@PreDestroy`).

The idea is that other specifications (beginning with EJB, CDI, JSF, and the new Java Interceptors spec) build upon this base-component model and layer additional services, for example, transaction management, type-safe dependency injection, interceptors. So at this level, the managed beans, CDI, interceptors, and EJB specifications all work hand-in-hand and are highly complementary.

Now, the managed beans specification is quite open-ended with respect to identifying exactly which classes are managed beans. It does provide the `@ManagedBean` annotation as one mechanism, but it also allows other specifications to define different mechanisms.

Java Persistence API (JPA) 2

The framework for managing relational data in applications is, as you might know, called **Java Persistence API** or better **JPA**.

The following features can be expected within JPA 2.0:

- Improved Object/Relational mapping
- Type-safe criteria API
- Expanded and richer JPQL
- second-level cache

- New locking modes:
 - PESSIMISTIC_READ—grabs shared lock
 - PESSIMISTIC_WRITE—grabs exclusive lock
 - PESSIMISTIC_FORCE_INCREMENT—updates version
- Standard configuration options
- Even the JDBC driver has a JPA interface. You can use this option in your JDBC connection specifications:

```
javax.persistence.jdbc.[driver | url | user | password]
```

See the following example:

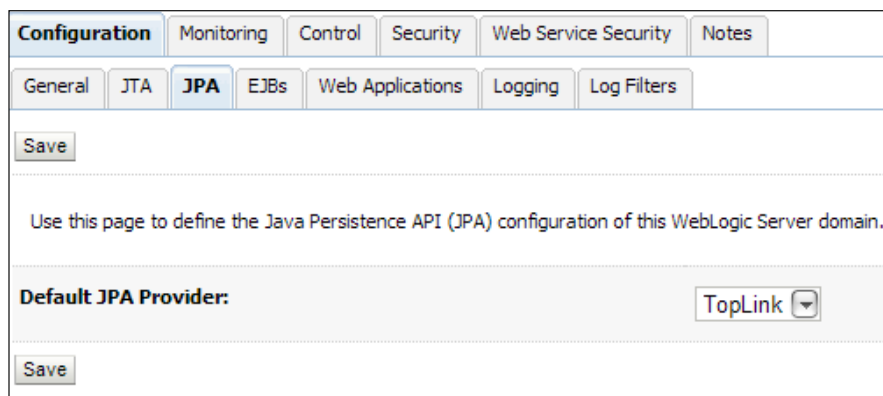
```
<property name="javax.persistence.jdbc.url" value="jdbc:oracle://localhost:3306/test2" />
<property name="javax.persistence.jdbc.driver" value="oracle.jdbc.OracleDriver" />
<property name="javax.persistence.jdbc.user" value="test" />
<property name="javax.persistence.jdbc.password" value="test" />
```

Some of the Java EE 6 specification are already supported in WebLogic 11g. JPA 2.0 is one of them. Though version 1.0 was the default 2.0 also worked.

Unless an explicit `<provider>...</provider>` is specified in the `persistence.xml` file of a deployed application, WebLogic 11g uses OpenJPA/Kodo by default. The default JPA provider setting is exposed via a new MBean: `JPAMBean` on the `DomainMBean`, and persists the configuration into the `config.xml` file.

Furthermore, you needed to install the patch QWG8 - Enable JPA 2.0 support on WebLogic Server.

To make it work on 11g, you had to use Oracle TopLink as the persistency provider as shown in the following screenshot:



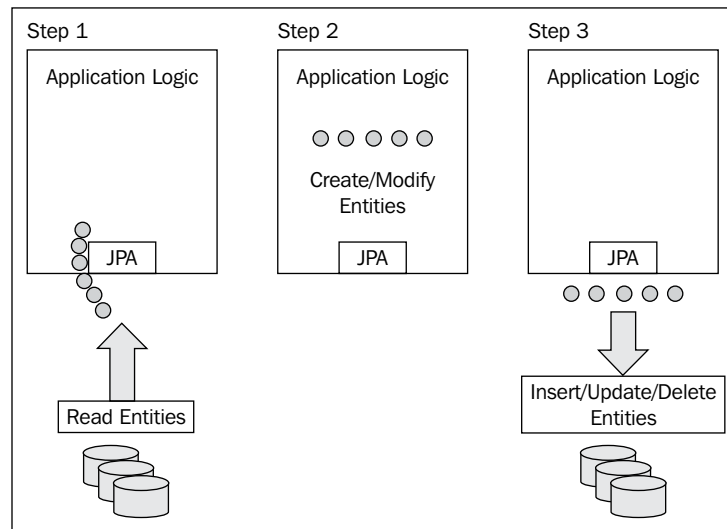
Also specify the provider in the <provider> element for a persistence unit in the persistence.xml file, for example:

```
<persistence-unit name="example">
  <provider>org.eclipse.persistence.jpa.PersistenceProvider</
provider>
...
</persistence-unit>
```

Now for WebLogic 12c, of course, JPA 2.0 is the default.

A very interesting feature to use is WebLogic Server with TopLink Grid, using JPA with Coherence as Level 2 Cache.

The following diagram explains how JPA works in an application:



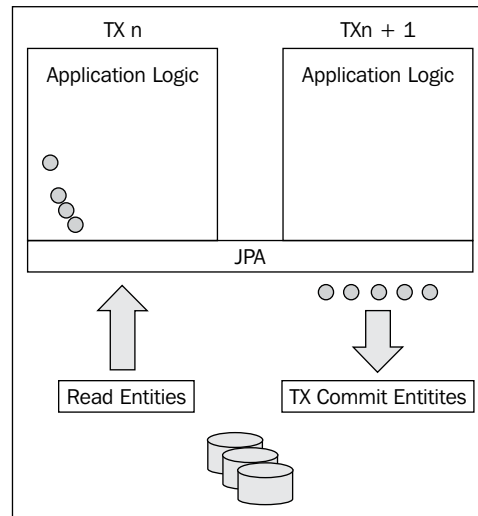
In this traditional way, no caching has taken place. With the use of JPA with coherence (for more details about Coherence, refer to *Chapter 6, Oracle WebLogic 12c to the Cloud: Exalogic*) you introduce a robust and high performing caching mechanism.

The use of a shared cache (L2) cache can improve application throughput.

TopLink Grid combines the simplicity of application development using the Java standard Java Persistence API (JPA).

EclipseLink JPA applications are using Coherence as a shared (L2) cache replacement along with configuration for more advanced usage. L2 caching is a sort of CPU caching of multicore processors, and this technique is being used by Coherence.

This following diagram shows you a basic example of JPA using caching mechanism:



Servlets 3.0

Java servlets, are components that execute on the server, accepting client requests and generating dynamic responses, and building dynamic content for web-based applications unlike some previous releases of the specification, which were just maintenance releases. The Servlet 3.0 specification is packed with lots of exciting features required for the new era of web development. The new specification focuses on delivering the following new features:

- Ease of development
- Pluggability and extensibility
- Asynchronous support with `Asynchronous Servlets@WebServlet(asyncSupported=true)`
- Security enhancements
- Other miscellaneous changes
- `@WebServlet`, `@WebListener`, `@WebFilter`
- Plugin libraries using web fragments
- Dynamic registration of Servlets
- `WEB-INF/lib/[*.jar]/META-INF/resources` accessible in the root
- Programmatic authentication login/logout
- Default error page

It is quite evident that servlets have enjoyed much wider use than any other technology in the Java Enterprise Edition family. The beauty of servlets remains in their simplicity and ability to process HTTP requests and pass the response back to web clients. Servlets can be used for implementing business logic of simple and small applications. In the case of web frameworks, servlets serve as an entry point (a controller servlet) for all incoming requests. Consequently, all popular frameworks are built on top of raw servlets. The new features in Servlet 3.0 are aimed at easing the development of servlet applications and will benefit both servlet developers and framework developers. In the following sections, let us look into each of these features in detail and see how we can utilize them for developing better applications.

Java API for RESTful Web Services (JSR 311)

RESTful Web Services, as you all might know do not need any additional messaging layer to transfer its data over an interface like HTTP, other than SOAP. It has a built-in set of rules from which you can create stateless services, to be viewed as resources. They can be identified by a unique URI.

Some of the changes that are made in Java EE 6 are:

- Web services through REST instead of SOAP
- REST counterpart of JAX-WS
- Gets rid of low-level code so you can focus on core logic
- Annotations from the ground-up
- Integrated with CDI and EJB

Other than 11g, 12c supports RESTful Web Service Development on WebLogic Server. It supports Jersey 1.9 JAX-RS Reference Implementation (RI), which is a production quality implementation of the JSR-311 JAX-RS 1.1 specification.

JAX-RS can also be integrated with EJB Technology and CDI:

```
@Stateless
@Path("/webservices")
public class PacktPublisherBean {
    @PersistenceContext
    private EntityManager entityManager;
    @PUT
    @Path("/packt/{publisher}")
    public void packtPublisher(
        @PathParam("publisher")
        String publisher,
```

```
@QueryParam("book")
String book,
@QueryParam("title")
Double bookPrice) {
    entityManager.persist(
        new Bid(publisher, book, bookPrice));
}
```

Java EE Connector Architecture 1.6

As known, JCA contains resource adapters which are software components that allow Java EE application components to access and interact with the underlying resource manager of the EIS.

WebLogic Server supports several ease-of-development and ease-of-configuration features introduced in the Java EE Connector Architecture version 1.6, including the following:

- **Metadata annotations** — use the annotations in resource adapter class files. Annotations can specify deployment information, eliminating the need to create the `ra.xml` deployment descriptor file manually.
- Dynamic configuration properties that can be defined on `ResourceAdapter`, `ManagedConnectionFactory`, and Admin Object beans.
- The ability to specify, at runtime, the transaction support a resource adapter can provide

Deprecated APIs

The following interfaces are no longer supported in Java EE 6 and are deprecated:

- **JAX-RPC**: This interface was previously superseded by JAX-WS, although still used in Java EE 5
- **EJB 2.x Entity Beans CMP**: Dropped in as an offer for JPA
- **JAXR**: UDDI not well used
- **Java EE Application Deployment (JSR-88)**: Development specification provided poor support

WebLogic 12c shared libraries and modules

In the previous sections of this chapter, we discussed the new Java EE 6, but how do we get it to work under our new WebLogic 12c Server?

In WebLogic Server 12c, you can make use of the shared Java EE library feature in WebLogic Server which provides an easy way to share one or more different types of modules among multiple enterprise applications. A shared library is a single module or collection of modules that is registered with the Java EE application container upon deployment. A shared library could be:

- Standalone EJB module
- Standalone web application module
- Multiple EJB modules packaged in an enterprise application
- Multiple web application modules packaged in an enterprise application
- Single plain JAR file

You should package a shared library into your built and compiled archive file (EAR, JAR, or WAR). However, you may also choose to deploy shared as exploded archive directories to facilitate repeated updates and redeployments (for development purposes).

After the library has been registered, you can deploy applications which refer to the library. Each referencing application receives a reference to the required library on deployment, and can use the modules that make up the library as if they were packaged as part of the referencing application itself. The library classes are added to the classpath of the referencing application, and the referencing application's deployment descriptors are merged (in memory) with those of the modules that make up the shared Java EE library.

There are some things to keep in mind while using shared libraries. If you develop shared Java EE libraries and optional packages, follow these rules:

- You should use shared libraries to share them amongst one or more Java EE modules (EJBs, web applications, enterprise applications, or Java classes) with multiple applications.
- If you need to deploy a standalone Java EE module, such as an EJB JAR file, as a shared Java EE library, package the module within an enterprise application. This avoids potential URI conflicts, because the library URI of a standalone module is derived from the deployment name.

- Optional packages are used when multiple Java EE archive files need to share a set of Java classes.
- If you have a set of classes that must be available to applications in an entire domain which do not need to be updated very frequently, then use the domain `/lib` subdirectory instead of using the shared Java EE libraries or optional packages. Classes in the `/lib` subdirectory are made available (within a separate system-level classloader) to all Java EE applications running on WebLogic Server.
- Always use versioning, even if you do not intend to enforce version requirements with dependent applications. Specifying versions for shared libraries enables to deploy multiple versions of the shared files for testing.
- It is better to specify an `Extension-Name` value for each particular shared library. If you don't, one will be taken from the deployment name of the library. Default deployment names are different for archive and exploded archive deployments, and they can be set to arbitrary values in the deployment command.
- For web applications, always use a unique context root when you develop it as a shared Java EE library. If the context root conflicts with the context root in a dependent Java EE application, use the `context-root` element in the EAR's `weblogic-application.xml` deployment descriptor to override the library's context root.
- For your deployers and admins, create a package of the shared libraries, package them into an archive file and deploy libraries from exploded archive directories during development to allow for easy updates and repeated redeployments.
- Deploy and target shared Java EE libraries to all WebLogic Server instances on which you want to deploy dependent applications and archives, otherwise the application fails.

Java classes compatibility

One of the nice things about the new WebLogic 12c is that you don't have to recompile or rebuild its previous application classes or libraries.

With one exception, upgrading to WebLogic Server 12c Release 1 (12.1.1) does not require you to recompile applications in order to create new generated classes.

The only exception here is the EJBGen utility. The current version of the EJBGen utility recognizes only JDK 5.0 or later metadata annotation-style EJBGen tags and not the old Javadoc-style tags. This means that source files that use the Javadoc-style tags must be upgraded to use the equivalent annotation, and then recompiled using the updated version of EJBGen.

So how does Java EE 6 fit in to Cloud Technology?

The WebLogic 12c has been utilized for Oracle technical solutions for the cloud, Engineered Systems or better called Exalogic. Applications running in the cloud with the Java EE S6 specifications have:

- Tighter requirements for resource and state management
- Better isolation between applications for controlling lifecycling in the Java Virtual Machine
- Standard APIs for NoSQL Databases, Caching, and so on
- Support for HTML5
- Common management and monitoring interfaces
- Better packaging

Summary

This chapter was focused on the new Java EE 6 and how it interacts with the new Oracle WebLogic 12c. Of course, this chapter or even this book is too short to discuss them all, but it will give you a good view about how it's implemented in a typical Java EE application, which of course Oracle WebLogic 12c is.

Some key facts about Java EE 6 to remember are:

- Designed for the full or lightweight Web 2.0 profile
- Innovation: New APIs, new features, further ease-of-use with managed beans, CDI, JSF 2, EJB 3.1, JPA 2, Servlet 3, JAX-RS, and bean validation
- The goal is to deprecate APIs that are outdated

With focus on the Cloud, it offers:

- Containers
- Injectable services
- Scale to large clusters
- Security model

In the next chapter, we will see how these applications will land in WebLogic Server.

3

Deployment, Installation, and Configuration Features

Once we have looked at WebLogic 12c and how it fits into the Java EE 6 techniques, let's have a closer look at how we can deploy, build, integrate, and configure our applications into the new WebLogic 12c.

Besides getting it certified for JAVA EE 6, Oracle has done a great job making it ready for the Cloud, Oracle's own engineered systems or Public Cloud.

Nevertheless, WebLogic 12c also runs perfectly on conventional systems, using all the built-in optimizations and benefitting from all the new built-in enhancements. WebLogic 12c has been optimized for deployment and running on:

- Oracle's Private Cloud, Exalogic
- Public Cloud
- Conventional systems

In this chapter we will have a closer look in the diverse tools and interfaces for development, deployment, upgrades, and other new configuration features.

Develop, build, compile, and deploy on WebLogic 12c

Developing an application is quite a complicated process from the first version of an application until it becomes live and is in production status. WebLogic 12c provides you the best help and support to make it an easy, quick but also reliable process with finally the desired results.

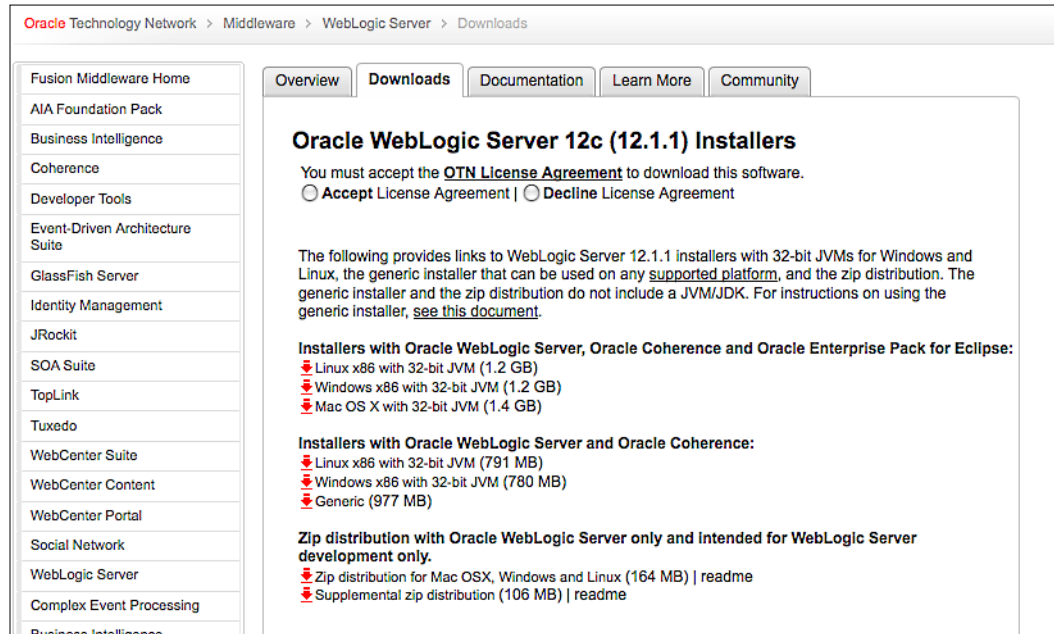
Lightweight development with WebLogic 12c

Just for pure development, Oracle provides a lightweight ZIP file installation, especially meant for quick building of an environment. This is a complete WebLogic Server installation in a ZIP file for development use only and supported on Windows, Linux, and Mac OS X systems. The extracted installation contains all the necessary artifacts you need to develop applications on WebLogic Server, but uses less disk space than a normal WebLogic Server distribution. It includes all WebLogic Server components *except* for the following:

- Samples
- Derby database
- Webserver plugins
- Native JNI libraries for unsupported platforms
- Administration Console help files for non-English locales
- Oracle Configuration Management (OCM) and ADR files
- Sun SDK and Oracle JRockit SDK
- Coherence libraries

For developers, this is benefit, because you just unzip it, create a WebLogic Domain, start it, and within a few minutes you will have it running, ready for your applications. The supplementary ZIP file provides you the samples and the Derby database in case you might need it.

To download the ZIP installer, go to **Oracle Technology Network**, there you will see a link for ZIP distribution.



Some hints and tips using development on WebLogic 12c

To use and configure your development environment as optimally as you can, consider the following hints:

Using FastSwap

When you want to use the `FastSwap` option, please think about the following best practices:

- In the `weblogic-application.xml` descriptor, `FastSwap` should be enabled in your exploded deployment archive
- The compiled source code modules should be in their corresponding directories in the exploded deployment structure
- Do not use archives, must be unpackaged classes

- Deploy application to WebLogic Server using the exploded deployment structure
- Configure your favorite IDE to directly compile classes in the exploded deployment structure
- In your IDE, you should develop, edit, and autocompile immediately

```
<?xml version="1.0" encoding="UTF-8"?>
<weblogic-web-app
  xmlns="http://xmlns.oracle.com/weblogic/weblogic-web-app"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd http://xmlns.oracle.com/weblogic/web
  <jsp-descriptor>
    <keepgenerated>true</keepgenerated>
    <debug>true</debug>
  </jsp-descriptor>
  <context-root>/PetClinicJavaEEContextPath</context-root>
  <fast-swap>
    <enabled>true</enabled>
  </fast-swap>
</weblogic-web-app>
```

Using the wlx option

Standard WebLogic Server starts up with the wls option. The default is wls, which starts all the WebLogic Server services, including EJB, JMS, connector, clustering, deployment, and management.

The wlx option starts a server instance that excludes the following services, making for a lighter weight runtime footprint:

- Enterprise JavaBeans (EJB)
- Java EE Connector Architecture (JCA)
- Java Message Service (JMS)

Using WebLogic server tooling

WebLogic Server provides a wide variety of helpful tooling to help developers.

- Incorporate into development process as necessary
- Before using it, set environment before executing
<DOMAIN_HOME>/bin/setDomainEnv.cmd (sh)

For several types of applications, there are tools available as follows:

- weblogic.appc: Compiles JSPs, EJB, and validates deployment descriptors
- weblogic.marathon.ddinit.EarInit: Generates EAR-level deployment descriptors
- weblogic.marathon.ddinit.WebInit: Generates web module deployment descriptors

- `weblogic.DDConverter`: Converts deployment descriptors to current WLS version
- `weblogic.Deployer`: Command-line deployment utility
- `weblogic.PlanGenerator`: Generates a template deployment plan for an application

Standard Java IDE support

To support developers in their work, WebLogic 12c supports a few IDEs to develop, build, compile, and deploy their Java applications onto WebLogic Server. WebLogic 12c supports the following IDEs.

Eclipse and Oracle Enterprise Pack for Eclipse (12.1.1.0)

In November 1998, IBM began creating a development tool platform that eventually became known as Eclipse, as a new Java IDE to create new products and applications.



Eclipse was based on an open, free platform, but that base would be complemented by commercial companies encouraged to create for-profit tools built on top of the Eclipse platform. Most of the committers and contributors to Eclipse came from a short list of the commercial vendors, with IBM being the largest contributor.

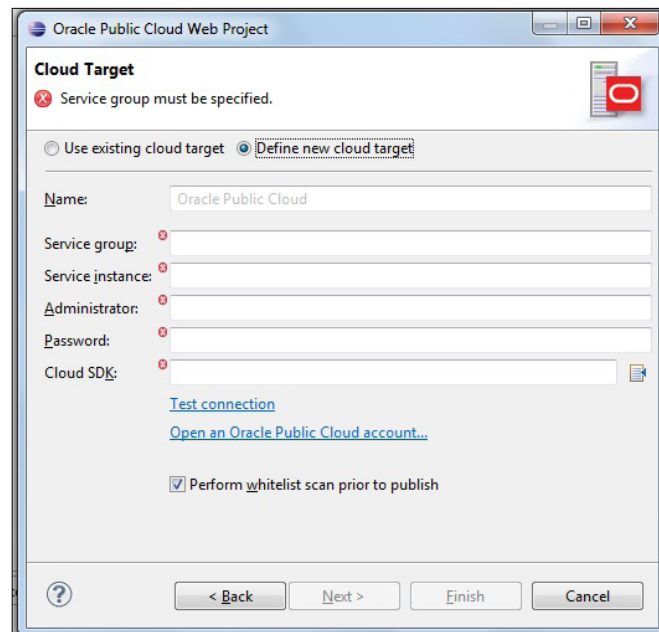
Also, Oracle is now one of the contributors with its Enterprise Pack for Eclipse, used as a set of plugins designed to support Java EE development, if you use Eclipse as your IDE.

At the time of writing this book, Oracle hasn't released JDeveloper 12c, OSB 12c, or SOA Suite 12c is not yet available on WebLogic 12c, but in this case, you can use Eclipse in combination with the Oracle Enterprise for Eclipse 12.1.1 (OEPE), so you still can build WebLogic SCA application and deploy it on WebLogic 12c.

To build SCA applications in WebLogic, use WAR located at `wlserver_12.1\common\deployable-libraries\weblogic-sca-1.1.war` as a shared library and target it to the server instance you want to deploy your SCA application to.

Oracle Enterprise Pack for Eclipse 12c provides the following key features:

- Oracle Application Development Framework Support, a set of plugins to create, configure, and run Oracle ADF applications.
- WebLogic Server support. You can:
 - Develop applications faster with virtual EAR technology
 - Deploy applications remotely
 - Edit deployment descriptors and plans
 - Support for WebLogic Shared Library
 - Support for WebLogic SCA
 - Support for WLST and python syntax
 - Support for XMLBeans
 - You can use EJBGen
 - Support for Web services
- Support for Oracle's Public Cloud. You can deploy your applications to the Cloud. You can specify to target your applications to the Cloud.



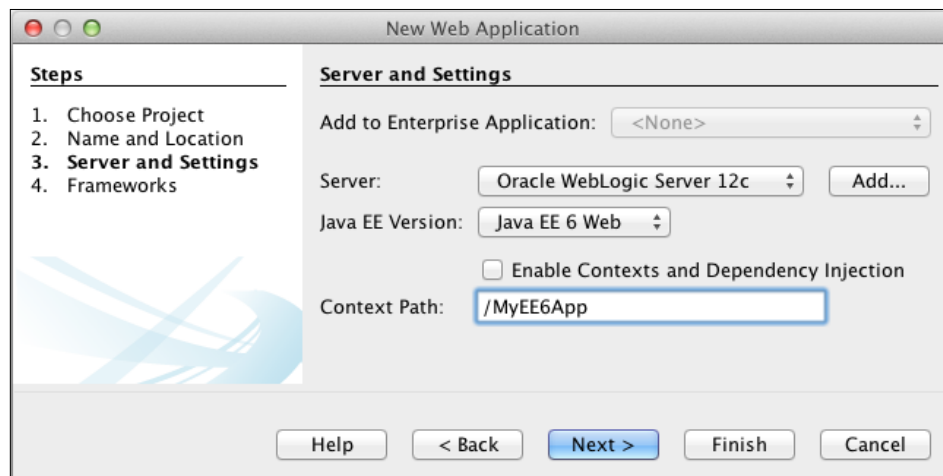
- Support for Web Services. Use standard Web Service technologies, such as XML, SOAP, and WSDL.
- Oracle Database Support lets you connect and query Oracle Databases.
- Support for Object Relational Mapping, Spring, and Web Application Development, using a technology called **AppXray** for analyzing the JSP pages, Java source files, resource bundles, and web configuration files.
- Support for coherence.

NetBeans IDE 7.1

Another well-known IDE for developing Java EE applications is NetBeans. NetBeans IDE is a free, open source IDE, written in Java and can run on Windows, Mac OS, Linux, Solaris, and other platforms supporting a compatible JVM. The JDK comes bundled with the release.

NetBeans 7.1 provides support for WebLogic 12c. The IDE works in fact with any standard Java Enterprise Edition (Java EE) container, such as GlassFish Server Open Source Edition 3.1.1, WebLogic 12c and 11g, Apache Tomcat, and JBoss.

The following screenshot shows you the creation of a new Web Application for deployment to WebLogic 12c:



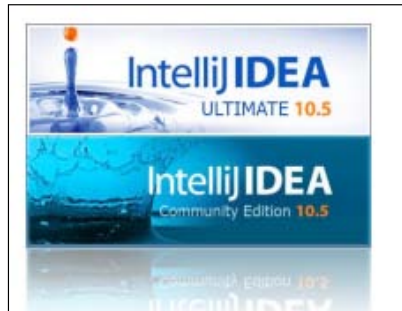
Other expected IDEs

You might expect JDeveloper 12c, but at the time of publishing this book this version was not yet available. Oracle will release with this version release during 2012, along with other Fusion Middleware Products which are not yet on the 12c release such as Oracle SOA/BPM Suite and Oracle Service Bus.

Another IDE that is coming up to support WebLogic 12c is IntelliJ's IDEA from JetBrains, and then the open Community Edition.

Some features that IntelliJ IDEA has are:

- Rich Code Editor understanding various code syntaxes. You can refactor and inspect your code. Navigation is good and clear. Also, full support for Java 7.
- Various test integrations such as JUnit and TestNG plus its own test runner UI.
- Diverse setups for projects and builds if you use Maven, Gradle, or Ant-based projects.
- Google Android development including latest SDK support.
- User interface and integration for version control systems, such as Subversion, Git/GitHub, Mercurial, and CVS.
- An XML editor with a built-in Java XML interface.
- If you are developing Java applications for desktops you can use the Swing UI designer.



WebLogic 12c and Maven integration

When working on a project, it is good to have a central system that controls and builds your software projects. One well-known tool for this from Apache is **Maven**.

So in general, Maven is:

- An automated build system
- A project management system
- A library and dependency handling system
- A project description system
- A site generation system

With Maven you can keep a set of standards, and maintain project lifecycling. You can define phases in your lifecycle like for instance when you'd like to execute a particular build or plugin. An important thing within Maven is the project object model.

This is well known to Maven users. If your project is using a well-defined project object model (POM), Maven can then apply cross-cutting logic from a set of shared or custom plugins.

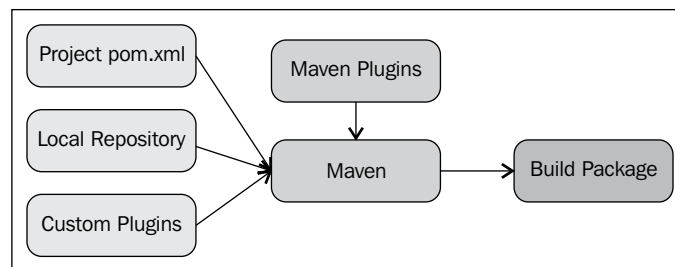
The project object model (POM)

One of the fundamentals in Maven is of course the POM. It describes all kinds of important data of a project such as the project, its name/version, type, and dependencies. It standardizes your configuration and standard directory layout for project, so you do not need to configure, and no path settings are required.

POM can automate building and packaging and it bundles all tests, resources, and classes.

With the POM you will have well-defined project life cycling.

In the following screenshot, you can see the Maven Build process:



Maven support was already introduced in 11g R1 PS3 (10.3.4) supporting Application Deployment operations with:

- Maven Mojo + WebLogic Deployer + WebLogic Client
- Supported Deployment Lifecycle operations: list-apps, deploy/undeploy, start, stop, and update

WebLogic 12c provides additional functionality since the 11g release like:

- Installation of Weblogic ZIP distribution onto a machine where WebLogic has not been installed
- WebLogic domain creation
- Start/Stop WebLogic Servers
- Execute WLST Scripts

You can use Maven installation to install the WLS maven plugin.

See how the WebLogic Maven plugin is installed:

```
C:\weblogic-12c-user-examples\> cd \maven-examples\simple-maven-example> mvn package
[INFO] Scanning for projects...
[INFO]
[INFO] Building simple-maven-example
[INFO] task-segment: [package]
[INFO]
[INFO] [resources:resources (execution: default-resources)]
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\weblogic-12c-user-examples\> cd \maven-examples\simple-maven-example\src\main\resources
[INFO] [compiler:compile (execution: default-compile)]
[INFO] Compiling 1 source file to C:\weblogic-12c-user-examples\> cd \maven-examples\simple-maven-example\target\classes
[INFO] [resources:testResources (execution: default-testResources)]
[WARNING] Using platform encoding (Cp1252 actually) to copy filtered resources, i.e. build is platform dependent!
[INFO] skip non existing resourceDirectory C:\weblogic-12c-user-examples\> cd \maven-examples\simple-maven-example\src\test\resources
[INFO] [compiler:testCompile (execution: default-testCompile)]
[INFO] No sources to compile
[INFO] [surefire:test (execution: default-test)]
[INFO] No tests to run.
[INFO] [war:war (execution: default-war)]
[INFO] Packaging webapp
[INFO] Assembling webapp [simple-maven-example] in [C:\weblogic-12c-user-examples\> cd \maven-examples\simple-maven-example\target\simple-maven-example-1.0]
[INFO] Processing war project
[INFO] Webapp assembled in [24 msecs]
[INFO] Building war: C:\weblogic-12c-user-examples\> cd \maven-examples\simple-maven-example\target\simple-maven-example-1.0.war
[INFO]
[INFO] BUILD SUCCESSFUL
[INFO]
[INFO] Total time: 1 second
```

A typical `pom.xml` for the WebLogic Maven plugin can be seen in this line of XML code:

```
<plugin>
<
groupId>com.oracle.weblogic</groupId>
<plugin>
<
groupId>com.oracle.weblogic</groupId>
gp g gp
<artifactId>wls-maven-plugin</artifactId>
```

```

<artifactId>wls-maven-plugin</artifactId>
<version>12.1.1.0</version>
<configuration>
<middlewareHome>c:\wls1211</middlewareHome>
<adminurl>t3://127.0.0.1</adminurl>
<build>
<plugins>
<build>
<plugins>
<version>12.1.1.0</version>
<configuration>
<middlewareHome>c:\wls1211</middlewareHome>
<adminurl>t3://127.0.0.1</adminurl>
<user>weblogic</user>
<password>welcome1</password>
<user>weblogic</user>
<password>welcome1</password>
</configuration>
</plugin>
</configuration>
</plugin>
</plugins>
</plugins>
</build></build>

```

One of the nice features is that you can manipulate your WebLogic domain by stopping, starting, deploying, undeploying, and redeploying.

With a simple Maven command, you can start the domain:

```
mvn wls:start-server
```

```

C:\weblogic-12c-user-examples\maven-examples\weblogic-provisioning-example>mvn wls:start-server
[INFO] Scanning for projects...
[INFO]
[INFO] Building weblogic-provisioning-example
[INFO] task-segment: [wls:start-server] (aggregator-style)
[INFO]
[INFO] [wls:start-server {execution: default-cli}]
[INFO] ++++++
[INFO] ++ wls-maven-plugin: start-server ++
[INFO] ++++++
[INFO] Starting server in domain: C:\weblogic-12c-user-examples\maven-examples\weblogic-provisioning-example\target\domain
[INFO] Check stdout file for details: C:\weblogic-12c-user-examples\maven-examples\weblogic-provisioning-example\target\domain\
server-5223679401799187816.out
[INFO] Process being executed, waiting for completion.
...
[INFO] Server started successful
[INFO]
[INFO] BUILD SUCCESSFUL
[INFO]

```

Advanced features of WebLogic Maven plugin

An advanced feature is automating deployment parameters using Maven properties.

You can bind the WebLogic plugin to Maven execution phases for automating deployment and provisioning operations. Also, you can use Maven profiles for multiple deployment targets and simplify configuration with inheritance and integration with continuous integration servers.

Another advanced feature is to associate WebLogic tasks with Maven phases. The Maven plugin goals can be bound to a Maven phase to execute during that phase. This can be very useful for automating deployment of target application to a server for testing as part of Maven lifecycle. Use the `<executions>` section of the `<plugin>` tag to specify the target phase and the goal(s) to execute.

Here you can see the plugin for phase goals:

```
<executions>
  <execution>
    <id>deploy</id>
    <phase>pre-integration-test</phase>
    <goals>
      <goal>deploy</goal>
    </goals>
    <configuration>
      <source>${project.build.directory}/${project.build.finalName}.${project.packaging}</source>
      <name>${project.build.finalName}</name>
    </configuration>
  </execution>

  <execution>
    <id>undeploy</id>
    <phase>post-integration-test</phase>
    <goals>
      <goal>undeploy</goal>
    </goals>
    <configuration>
      <name>${project.build.finalName}</name>
    </configuration>
  </execution>
</executions>
```

Maven support for several IDEs

Maven provides support for IDEs used for WebLogic development.

Some common features:

- Materialize projects from Maven POM
- Dependency integration with Compile/Test/Run classpath
- Maven repository browsing
- Multimodule Maven project support
- Form-based POM Editor
- Dependency graph

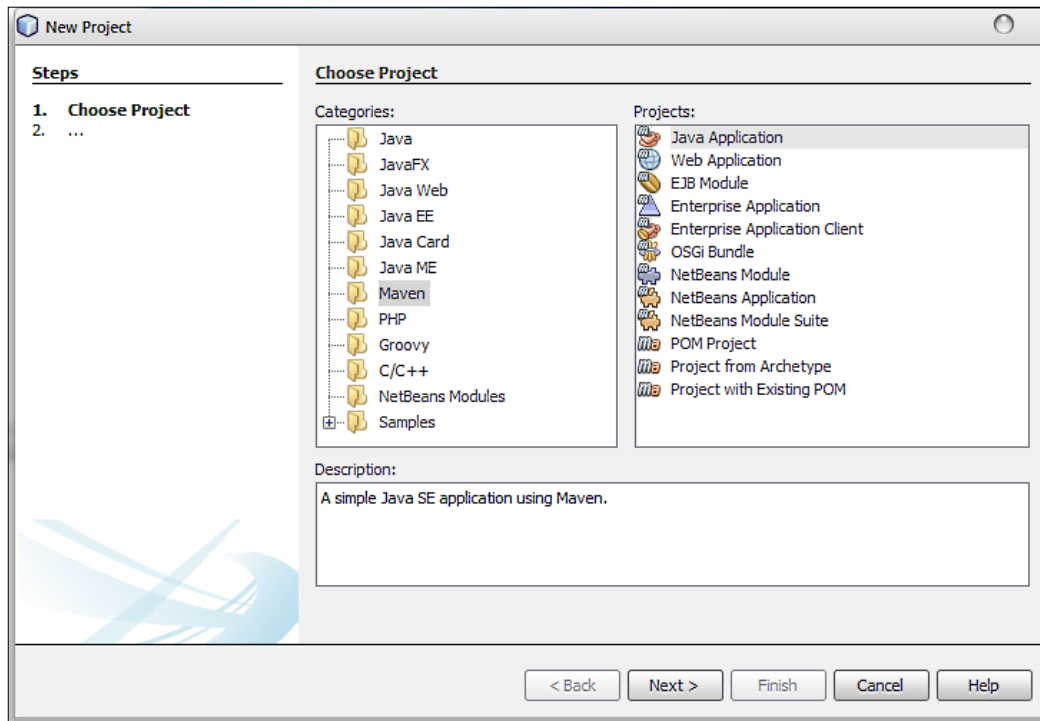
Maven for Eclipse/OEPE

To support Eclipse with Maven, you can download and install the open-source plugin called `M2Eclipse` from Sonatype. With this plugin, you can do the following things:

- Create and export Maven projects
- Dependency management and integration with the Eclipse classpath
- Automatic dependency downloads and classpath updates
- Create projects with Maven Archetypes
- Browse and searching remote Maven repositories
- POM management with automatic update to dependency list
- Adapt nested multimodule Maven projects to the Eclipse IDE
- Form-based and text-based POM Editor
- GUI presentation of dependency tree and resolved dependencies

NetBeans and Maven

NetBeans 6.7+ provides native support for Maven so you can create or import Maven projects.



Some other features are as follows:

- Manage dependencies
- Dependency graph viewer
- Run/Test projects
- Maven repository browser
- Can also push a Maven job directly to a Hudson CI server as build job

Classloading and the Classloading Analysis Tool (CAT)

Always a matter of concern with developed applications is: how does the application start, and which classes, application, or server side will be loaded and in what order during startup phase. Also, conflicts between application classes and server classes can decrease application functionality or even make an application to start.

As a developer, you are responsible for developing and assembling applications, so make use of many sources of code/libraries within applications.

- Use in-house libraries + Spring, Xerces, Log4J, apache-commons-*, and so on
- Understanding how class loading works is important to make correct and efficient use of Java libraries
- What classes get loaded from where
- Efficiently reusing shared libraries
- Avoiding `ClassCastException`s
- WebLogic Server class loading is a powerful mechanism that can be used to good effect
- Reuse of common shared libraries
- Filtering `ClassLoader` to control library visibility
- Construction of custom module `ClassLoader` hierarchies

Overview of Java EE application Classpath

Within the Java EE implementation standards, you can identify the following classpaths:

EAR application classpath:

- `APP-INF/classes/`
- `APP-INF/lib/*.jar`
- Manifest classpath:
 - `(EAR-library-classpath) *`
 - `(JAR-library-classpath) *`

WAR application classpath:

- `WEB-INF/classes/`
- `WEB-INF/*.jar`

- Manifest classpath:
 - (WAR-library-classpath) *
 - (JAR-library-classpath) *
 - (EAR-Application-classpath)

To avoid Classloader conflicts, use filtering. This enables classes to be loaded from an application-supplied library first. Classloading filtering has the following features:

- Changes the delegation model from parent to child first
- Works as a barrier to prevent parent from supplying class
- Does not load classes itself
- Useful in scenarios where application needs to use a different version of a framework that is already bundled with the server
- Xerces, Spring, Ant, Commons-Logging

The following is an application with a list of packages in the deployment descriptor:

```
<weblogic-application>
...
<prefer-application-packages>
<package-name>org.apache.xerces.*</package-name>
<package-name>org.apache.commons.*</package-name>
<package-name>org.apache.log4j.*</package-name>
</prefer-application-packages>
...
</weblogic-application>
```

Built-in WLS CAT (ClassLoading Analysis Tool)

Starting from WebLogic 10.3.4, the WLS-CAT tool was a built-in tool for analyzing classes from the Classloader configuration of an application. It is web-based and you can easily detect Classloading issues, doing some debugging in an application's classpath, and even giving clues how to resolve it. It's a standalone web application, located in the WebLogic Server Home, `/server/lib/wls-cat.war`, and it has a reference to every deployed application.

Settings for NewsApp	
Overview	Deployment Plan
Configuration	Security
Targets	Control
Testing	Monitoring

Some deployment types support test points you can use to verify that a deployment was successful and that the following table includes all of the test points available for this application or module.

Deployment Tests

Name	Test Point
NewsApp	
/NewsApp-war	
/wls-cat/index.jsp	Classloader Analysis Tool
NewMessage	
NewsEntityFacade	

By clicking on the link, WLS-CAT starts up for that specific application and gives you details about the application classes and possible conflicts. WLS-CAT analyzes classes loaded by the system classpath Classloader and the WebLogic Server main application Classloaders. You can perform analysis at the class, package, or JAR level.

WLS-CAT could tell you to use the web app Classloader by setting the preferred-application-packages in your `weblogic.xml` as specified in the preceding code.

The following is the detailed view of a Classload analysis:

Classloader Tree for Application: NewsApp

View: basic | detailed
Actions: Summary | Analyze Conflicts | **Classloader Tree** | Generate Report

Information

Application Info

- Application: NewsApp
- Version: <Not Set>
- Annotation: NewsApp@

System Classloaders

Type: sun.misc.Launcher\$ExtClassLoader
HashCode: 13288040
Classpath:
/C:/Oracle/Middleware/jdk160_29/jre/lib/ext/dnsns.jar
/C:/Oracle/Middleware/jdk160_29/jre/lib/ext/localedata.jar
/C:/Oracle/Middleware/jdk160_29/jre/lib/ext/sunjce_provider.jar
/C:/Oracle/Middleware/jdk160_29/jre/lib/ext/sunmscapi.jar
/C:/Oracle/Middleware/jdk160_29/jre/lib/ext/sunpkcs11.jar

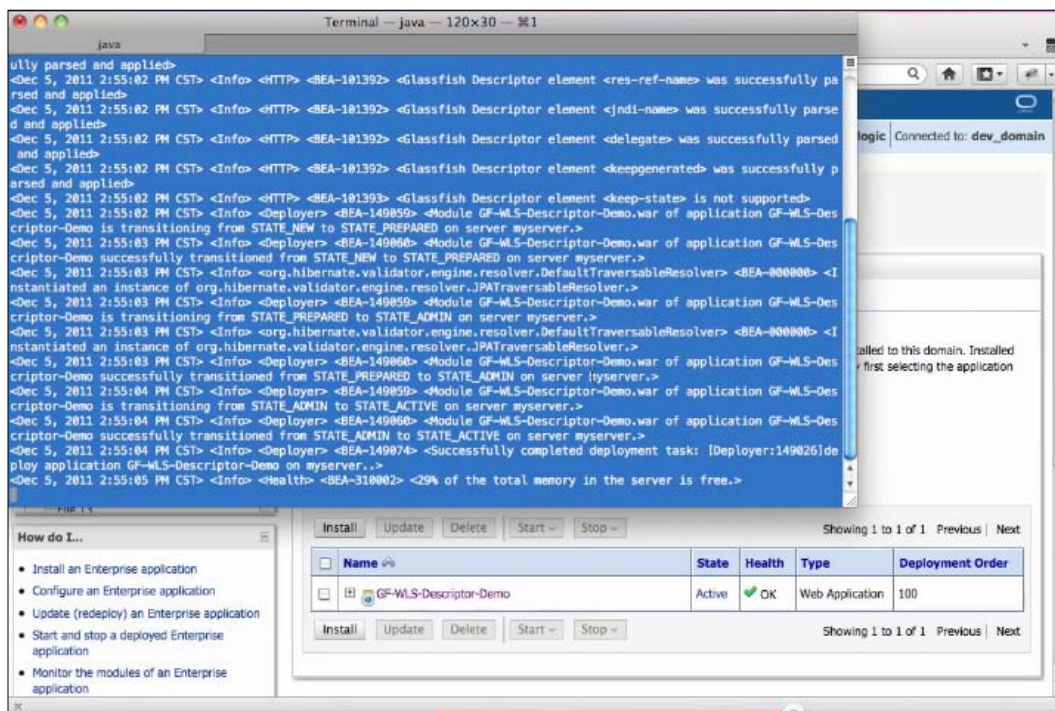
Type: sun.misc.Launcher\$AppClassLoader
HashCode: 16032330
Classpath:
/C:/Oracle/Middleware/jdk160_29/lib/tools.jar
/C:/Oracle/Middleware/modules/com.bea.core.redefagent_2.0.0.0.jar
/C:/Oracle/Middleware/modules/features/weblogic.server.modules_12.1.1.0.jar
/C:/Oracle/Middleware/modules/net.sf.antcontrib_1.1.0.0_1-0b2/lib/ant-contrib.jar
/C:/Oracle/Middleware/modules/org.apache.ant_1.7.1/lib/ant-all.jar
/C:/Oracle/Middleware/patch_ocp371/profiles/default/sys_manifest_classpath/weblogic_patch.jar
/C:/Oracle/Middleware/patch_oepe100/profiles/default/sys_manifest_classpath/weblogic_patch.jar
/C:/Oracle/Middleware/patch_wls1211/profiles/default/sys_manifest_classpath/weblogic_patch.jar

The WLS-CAT is a very powerful and smart tool which gives you more knowledge about Classloading hierarchies, conflicts, and solutions.

Deployment descriptor support for GlassFish Server

GlassFish Server offers some support for `weblogic-application.xml`, `weblogic.xml`, and `weblogic-webservices.xml` deployment descriptor files. WebLogic Server 12c is able to easily deploy web applications developed for GlassFish, reading the `glassfish-web.xml` descriptor, and automatically applying a set of common configuration elements to the web application when it is running on WebLogic Server. The element in `weblogic-application.xml` that GlassFish Server supports is security. The equivalent element in the `glassfish-application.xml` file is security-role-mapping. As a consequence, an application that has been developed for Glassfish and deployed on that environment can easily be migrated on a Weblogic platform by a simple re-deploy. Of course, this is a subject for discussion depending on the application's security model since that will have to be accommodated for the Weblogic Security Framework.

See GlassFish redeployment to WebLogic in action:



Cloud development with WebLogic 12c

As been said before, the *c* in WebLogic Server 12*c* stands for cloud. WebLogic Server 12*c* is optimized to run as a high-performance, mission-critical, elastic-cloud infrastructure on the Exalogic Cloud. The Exalogic Cloud is tested and tuned to provide the best foundation for Java applications, Oracle applications, and other enterprise applications.

The Exalogic Elastic Cloud is a component of Oracle's Cloud Application Foundation, the company's next-gen app infrastructure. (Oracle calls it *the world's first and only engineered system for cloud computing*). It combines the Exalogic cloud and the WebLogic Server with Tuxedo for C/C++/COBOL, Oracle's Coherence in-memory data grid, the JRockit and Hotspot Java VMs, Oracle Enterprise Manager, and the Oracle Virtual Assembly Builder.

This release is part of Oracle's Java Cloud Service. The Java Cloud Service supports development and deployment from multiple Java-based integrated development environments (IDEs), including Oracle's own JDeveloper, its open-source NetBeans IDE and the Eclipse environment.

Installation and upgrades with WebLogic 12c

When WebLogic 12*c* was announced on December 1, 2011, a few days later the new release was available at OTN. Developers, administrators, and other technicians started downloading it to play around.

Oracle Technology Network > Middleware > WebLogic Server > Downloads

Fusion Middleware Home
AIA Foundation Pack
Business Intelligence
Coherence
Developer Tools
Event-Driven Architecture Suite
GlassFish Server
Identity Management
JRockit
SOA Suite
TopLink
Tuxedo
WebCenter Suite
WebCenter Content
WebCenter Portal
Social Network
WebLogic Server
Complex Event Processing
Business Intelligence

Overview Downloads Documentation Learn More Community

Oracle WebLogic Server 12c (12.1.1) Installers

You must accept the [OTN License Agreement](#) to download this software.
☒ Accept License Agreement | ☐ Decline License Agreement

The following provides links to WebLogic Server 12.1.1 installers with 32-bit JVMs for Windows and Linux, the generic installer that can be used on any [supported platform](#), and the zip distribution. The generic installer and the zip distribution do not include a JVM/JDK. For instructions on using the generic installer, [see this document](#).

Installers with Oracle WebLogic Server, Oracle Coherence and Oracle Enterprise Pack for Eclipse:

- Linux x86 with 32-bit JVM (1.2 GB)
- Windows x86 with 32-bit JVM (1.2 GB)
- Mac OS X with 32-bit JVM (1.4 GB)

Installers with Oracle WebLogic Server and Oracle Coherence:

- Linux x86 with 32-bit JVM (791 MB)
- Windows x86 with 32-bit JVM (780 MB)
- Generic (977 MB)

Zip distribution with Oracle WebLogic Server only and intended for WebLogic Server development only.

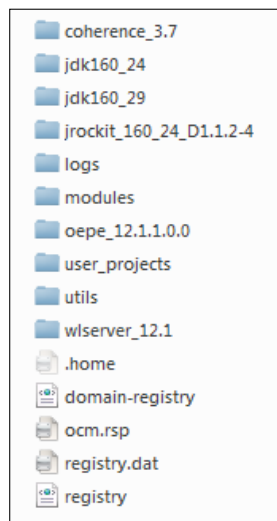
- Zip distribution for Mac OSX, Windows and Linux (164 MB) | [readme](#)
- Supplemental zip distribution (106 MB) | [readme](#)

The following types of WebLogic Server installers are available:

- OS-specific Package installer: This type of installer includes the JDKs for the selected platform. The installer can be an `.exe` (Windows) file or a `.bin` (Linux, Mac) file.
- Generic Package installer: This is a JAR file without the JRockit SDK and Sun JDK. You will need to pre-install Java for this type of installation.
- Upgrade installer: Upgrade installers allow you to upgrade an existing WebLogic Server installation to the current patch release. For example, if you have WebLogic Server 10.3.0 installed, you can use an Upgrade installer to upgrade your installation to WebLogic Server 10.3.6.

The installation process hasn't been changed from its prior release, and is pretty straightforward. Besides changing some splash-screens and version numbers, everything is still the same

Also after installation, the software directory structure seems pretty much the same, except of course, with updated software and version numbers.



Available Installers:

Filename	Description
wls1211_generic.jar	For installations on any supported platform on which Java is already installed. Includes WebLogic Server and Coherence, but without JDKs.
wls1211_linux32.bin	For installations on Linux x86 32-bit machines. Includes WebLogic Server, Coherence, Sun SDK 1.6_29, and Oracle JRockit 1.6_29.
wls1211_solaris32.bin	For installations on Solaris 32-bit machines. Includes WebLogic Server, Coherence, Sun SDK 1.6_29, and Oracle JRockit 1.6_29.
wls1211_win32.exe	For installations on Windows 32-bit machines. Includes WebLogic Server, Coherence, Sun SDK 1.6_29, and Oracle JRockit 1.6_29.
wls1211_dev.zip	Development-only installer that extracts a complete WebLogic Server installation.
wls1211_dev_supplemental.zip	Contains additional files that supplement the development-only installer (wls1036_dev.zip).

Upgrading to WebLogic 12c

Upgrades are available for WebLogic 8.1 up to the latest version 10.3.5. You can do the upgrade by using the patchset assistant or upgrade wizard. Also upgrades from IAS, GlassFish, and JBoss are supported.

Available upgrade strategies are:

- Internet Application Server: Automated tooling.
- WLS11g: simple upgrade. Use domain upgrade wizard if you are an existing WebLogic customer on WebLogic version 8, 9 or 10.X. It is not necessary for WebLogic Server applications to be undeployed. In most cases, WebLogic Server applications can be run without modifications in the new WebLogic Server 12c. Upgrade script can be found in `WL_HOME/common/bin`.
- Glassfish: simple redeploy.
- JBoss: migration services.

New configuration features in WebLogic 12c

At the launch of WebLogic 12c, Oracle told us there would be more than 200 new features available in this release. We won't discuss all 200 features but here are the most important ones. Except for supporting all the new JAVA EE6 features, the following features are also added or changed.

JDK 7 certification

Not immediately available but during February 2012 JDK7 support came out through an updated Oracle WebLogic Server 12.1.1 distribution. This distribution includes patches that enable Java SE Development Kit (JDK) 7 certification and provide other product optimizations. The patches can be found in the `MW_HOME/patch_wls1211/patch_jars`.

Administration Console

Several Admin Console changes have been made to support the implementation of Java EE 6, including changes to:

- Deployment
- Application container
- SCA container
- Split source
- Application and module naming

NodeManager

The default value for `startScriptEnabled` has been changed to `true` as of this release. In previous releases, the default was `false`.

JDBC

WebLogic 12c supports data sources using Java EE 6 specifications and provides an extended set of WebLogic data source configuration attributes.

The `name` element identifies a `Datasource` and is registered with JNDI. The value specified in the `name` element begins with a namespace scope:

- `java:comp`: Names in this namespace use single components
- `java:module`: Shared by all components in a module, for example, the EJB components defined in an `ejb-jar.xml` file

- `java:app`: Shared by all components and modules in an application, for example, the application-client, web, and EJB components in a `.ear` file
- `java:global`: Shared by all the applications in the server

WebLogic 12c has Active GridLink used as an optimization for RAC. It uses Fast Connection Failover for faster RAC failure detection.

Security

Security is a very important subject in today's IT systems. WebLogic 12c support, along with support for Java EE 6, also some new or enhanced security features. We will now look at some security features.

Resource Adapter security

WebLogic 12c adds support for the security context in the Admin Console creating inbound EIS-to-WebLogic principal mappings, which map EIS principals, such as users or groups defined in the EIS security domain, to corresponding principals in the WebLogic domain.

Java Authentication SPI for containers (JASPIC) support

The JASPIC specification defines a service provider interface (SPI) for authentication providers that use message authentication mechanisms, and can be integrated in server web application message processing containers or runtimes. This message processing runtime uses this SPI at the identified message processing points to delegate the corresponding message security processing to the authentication providers.

SSL

Some certificates are removed from 12c such as Certicom.

JSSE is the only SSL implementation that is supported in WebLogic Server 12.1.1. The default value for `JSSEEnabled` has been changed to `true`. Previous releases have it set to `false`. However, even if `JSSEEnabled` is set to `false`, it will be ignored. The `MBean` value will not be changed, either in memory or in the persisted `config.xml` file. WebLogic Server will continue to use JSSE, but will give a warning.

Standalone clients

The WebLogic Thin T3 Client JAR (`wlthint3client.jar`) supports GlassFish application server version 3.1 and higher. It is a lightweight alternative to the `wlfullclient.jar` and `wlclient.jar` (IIOP) remote client JARs. The Thin T3 client has a minimal footprint while providing access to a rich set of APIs that are appropriate for client usage. As you may expect, the Thin T3 Client uses the WebLogic T3 protocol, which provides significant performance improvements over `wlclient.jar`, which uses the IIOP protocol.

The Thin T3 Client is the recommended option for most remote client use cases. There are some limitations in the Thin T3 Client as outlined here. For those few use cases, you may need to use the full client or the IIOP thin client.

The Thin T3 Client can be used in standalone applications, and is also designed for applications running on foreign (non-WebLogic) servers. One common use case is integration with WebLogic JMS destinations.

Deprecated: `weblogic.management.username` and `weblogic.management.password`

In WebLogic 12c, the boot username and password system properties `weblogic.management.username` and `weblogic.management.password` have been deprecated and will be removed in a future release, and you will no longer be able to specify the username and password in the `startscript` for starting in production mode.

The only thing to use is `boot.properties`. `boot.properties` file should be created manually when running in the Production mode and should be placed in the Domain directory in the security folder of the Admin Server.

Web Services

Of course, Web Services in 12c are also Java EE6 compliant. For instance, it includes support for EJB 3.1. Specifically, WebLogic Web Services can be packaged as follows:

- EJB in a WAR file
- Singleton EJB

Also, some new samples for RESTful Web Service (JAX-RS) are added.

EclipseLink MOXy (JAXB) is used for data binding and JAXB provider. The EclipseLink MOXy component binds Java classes to XML schemas. MOXy implements JAXB to provide mapping information through annotations as well as providing support for storing the mappings in XML format.

The UDDI v2.0 registry and UDDIExplorer applications have been removed. Recommended is to consider using Oracle Enterprise Repository or Oracle Service Registry, which provide SOA visibility and governance.

Exalogic features

All specific Exalogic features, such as Oracle Virtual Assembly Builder, Exabus, and Oracle Traffic Director will be discussed in *Chapter 6, Oracle WebLogic 12c to the Cloud: Exalogic*.

To start using WebLogic 12c for Exalogic, you should set it in the Admin Console as shown in the following screenshot:

Home > WLS12c

Settings for WLS12c

Configuration | Monitoring | Control | Security | Web Service Security | Notes

General | JTA | JPA | EJBs | Web Applications | Logging | Log Filters

Save

A domain is a collection of WebLogic Server instances that is managed by a single Administration Server

* Indicates required fields

* **Name:** WLS12c

☐ **Enable Administration Port**

Administration Port: 9002

☐ **Production Mode**

☐ **Enable Exalogic Optimizations**

WebLogic 12c New feature TLog Store

In WebLogic pre-12c, the transaction logs were file-based only. In WebLogic 12c this has been enhanced with the possibility to store TLogs in a database. Here are the guidelines to achieve this:

First, create a JDBC data source. This can be any type (generic, multidata source, or GridLink data source), the only constraint being that it must neither be XA nor support Global Transactions.

Second, go to your server and select **Configuration | Services**.

Transaction Log Store

Type: Select "Default Store" if you want transactions logged to the server's default store. Select "JDBC" if you want transactions logged to a specific JDBC data source. [More Info...](#)

Data Source: The JDBC data source used by the transaction log store to log transactions. You cannot configure the transaction log store to use a JDBC data source that is configured to use an XA JDBC driver or configured to support global transactions. [More Info...](#)

Prefix Name: When using multiple TLOG JDBC stores, use this attribute to create a label ending in '_' that is prepended to the name of the server hosting the JDBC TLOG store and ends in '_' to form a unique JDBC TLOG store name for each configured JDBC TLOG store. [More Info...](#)

There's a new section here, **Transaction Log Store**. Switch the type to JDBC and specify your data source which should be in the drop-down list.

You can leave the prefix as is.

Summary

Over more than 200 features! Too many to discuss them all; some are minor updates, others are much more important. I hope this chapter has given you a good overview about the most important ones. Let's rush to the next chapter!

4

Integrated and External Services

In the previous chapters, we've seen WebLogic 12c's Java EE 6 readiness, we've seen how and what tools we can use to deploy our applications on WebLogic 12c, but now we have to dive into WebLogic Server itself to discover the exciting features it has in it.

At server side, WebLogic serves you a lot of interfaces to process the data. We will discuss them in this chapter, but also which new features a typical administrator should know before starting to use or upgrade to WebLogic 12c.

JDBC services

JDBC services have been enhanced significantly, with focus on availability and load balancing. We will dive into the new features supported in this release.

Active GridLink and RAC integration

Already available since version 10.3.4, a new feature for connection to RAC databases is Active GridLink for RAC. As you might know, RAC stands for Real Application Clusters and is a high-availability solution for Oracle databases. The database provides high availability in a cluster of RAC instances. Active GridLink for RAC and use of GridLink data sources are also available in this version as part of WebLogic Suite. This feature uses Oracle Notification Service (ONS) to detect states and changes of an Oracle RAC instance. This makes GridLink very scalable because the number of services in the cluster can be dynamically increased.

The following screenshot shows how to test a GridLink Datasource:

Create a New JDBC GridLink Data Source

Test All Listeners | Back | Next | Finish | Cancel

Test GridLink Database Connection

Test the database availability and the connection properties you provided.

What is the full package name of JDBC driver class used to create database connections in the connection pool?
(Note that this driver class must be in the classpath of any server to which it is deployed.)

Driver Class Name: oracle.jdbc.xa.client.OracleXADataSource

What is the URL of the database to connect to? The format of the URL varies by JDBC driver.

URL:

```
jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host1)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=gridlink)))
```

Click the test button to test each listener.

Test Listener jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host1)(PORT=1521)))(CONNECT_DATA=(SERVICE_NAME=gridlink)))

Test Listener jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=host2)(PORT=1522)))(CONNECT_DATA=(SERVICE_NAME=gridlink)))

Already in WebLogic 9.2, RAC was supported with Oracle 9i RAC, but was called a Multi Datasource configuration and didn't include all the features of Oracle RAC service configurations, you will need to create a separate multidata source for each defined service. The configuration is pretty static and needs manual intervention for adding or removing new Datasources.

Fan enabling

When creating a GridLink Datasource, you'll see the **FAN enabled** checkbox and a place to enter the ONS details. **FAN** is **Fast Application Notification**, which acts on certain application events in the database which will be used to balance the load. If FAN is not enabled, the default Round-Robin algorithm will be used.

This is enabled by default and the UI makes sure you enter at least one ONS server (of course, as FAN is not going to work otherwise). You'll also notice the wallet parameters which allow the ONS messages to be sent over SSL – usually recommended in production. A wallet is used for storing SSL Certificates.

Test All ONS Nodes
Back
Next
Finish
Cancel

Test ONS client configuration

Test the ONS client configuration properties you provided

Check to enable the data source to subscribe to and process Oracle FAN events

☒ **FAN Enabled**

Enter host and port of each ONS node separated by colon and click the add button.

ONS host and port:

onshost1:1521, onshost2:1521

Click the test button to test ONS node.

Test ONS Node onshost1:1521

Test ONS Node onshost2:1521

The location of the Oracle wallet file in which the SSL certificates are stored.

ONS Wallet File Directory:

<walletdir>

The wallet password attribute that is included as part of the ONS client configuration string. This attribute is only required when ONS is configured

ONS Wallet Password:

.....

Confirm ONS Wallet Password:

.....

New JDBC features for WebLogic 12c

This release includes the following new and changed features:

- The Capacity Increment Attribute:** WebLogic first uses the existing connections available in the pool when a connection request is issued. If there is no match, this parameter takes care of creating a new one.
- The MinCapacity Attribute:** The MinCapacity attribute sets the minimum number of physical connections that a connection pool can contain after it's initialized. The initial capacity that previously handled both the initial and minimum capacity for the pool has been split into two attributes:
 - MinCapacity defaults to InitialCapacity if not set; InitialCapacity continues to default to 1.
 - MinCapacity is only used for shrinking calculations. If you don't set MinCapacity, InitialCapacity is used.

- **Define Fatal Error Codes:** To indicate that the backend database is unavailable or unreachable on a connection, you can work with fatal error. The connection will be marked as invalid and taken out of the pool. The errors may include deployment errors that cause a server boot to fail, and connection errors.

You can specify this with an exception code within a `SQLException` (by `SQLException.getErrorCode()`), which indicates that a fatal error has occurred and the connection is no longer healthy and is to be removed from the connection pool. For Oracle databases, the following fatal error codes are already available within WLS and you don't have to configure them:

- 3113 – end-of-file on communication channel
 - 3114 – not connected to Oracle
 - 1033 – Oracle initialization or shutdown in progress
 - 1034 – Oracle not available
 - 1089 – immediate shutdown in progress - no operations are permitted
 - 1090 – shutdown in progress - connection is not permitted
 - 17002 – I/O exception
- **Data Source Profile Logging:** In previous versions, data source events were recorded as WLDF events. For better usability and performance, WebLogic Server now uses a data source profile log to store events. The profile log has log-rotation – the ability to configure, rotate, and retire old data.

You can see where to set your data source logging profile in the following screenshot:

Settings for AdminServer

Configuration Protocols **Logging** Debug Monitoring Control Deployments Services Sec

General HTTP **Data Source**

Save

Use this page to configure DataSource logging for the server. DataSource profile logging must be configured not store them in the server log file or the domain log file.

Log file name: logs/datasource.log

Rotation

Rotation type: By Size

Rotation file size: 500

Begin rotation time: 00:00

Rotation interval: 24

☒ **Limit number of retained files**

Files to retain: 7

- **Application-Scoped Drivers:** It is now possible to include a database driver in the EAR/WAR file that contains an application-scoped data source. You do not have to update the classpath of the manifest file to include the driver location.

Oracle BI Server Support: Just select **Oracle BI Server** as **Database Type** when creating a new generic data source to interoperate with the Oracle BI Server.

Database Type: Oracle BI Server

- **Keep Connection After Global Transaction:** This new feature enables WebLogic Server to keep a physical connection associated with a logical connection when committing or rolling back a global transaction. Defined as `KeepConnAfterGlobalTx` in `JDBCXAParamsBean`.

- **Session Affinity Policy:** Web applications where a user session has back-to-back OLTP should have better performance. In some cases, repeated operations against the same data sets are being processed at the same RAC instance. A GridLink data source uses the session affinity policy to improve performance by directing the database operations of a servlet session to the same RAC instance in an RAC cluster. A GridLink data source monitors RAC load balancing advisories (LBAs) using the `AffEnabled` attribute to determine if RAC affinity is enabled for an RAC cluster. The first connection request is load balanced using Runtime Connection Load-Balancing (RCLB) and is assigned an Affinity context. All the next connection requests will be transferred to the same Oracle RAC instance using the Affinity context of the first connection until the session ends or the transaction completes.
- **Connection Labeling:** Applications often initialize or re-initialize a connection, but with Connection Labeling, an application requests a connection with the desired label from the connection pool. By associating particular labels with particular connection states, an application can retrieve an already initialized connection from the pool without re-initialization. The `oracle.ucp.jdbc.LabelableConnection` interface is used to apply and remove connection labels, as well as retrieve labels that have been set on a connection. The `oracle.ucp.ConnectionLabelingCallback` interface will act when a labeled connection is requested but there are no connections in the pool that match already existing labeled connections.
- **New Debug Scopes:** Some new debug options are available like:
 - `weblogic.jdbc.rac.DebugJDBCUCP` – low-level UCP debugging. This includes both required and optional callback interfaces that are used to implement connection pool features, like the `ConnectionAffinityCallback` interface. This is used to create a callback that enables or disables connection affinity and can also be used to customize connection affinity behavior. You can set UCP debugging directly using:

```
oracle.ucp.level = FINEST;  
oracle.ucp.jdbc.PoolDataSource = WARNING;
```
 - `weblogic.jdbc.rac.DebugJDBCREPLAY` – REPLAY debugging.
 - `weblogic.jdbc.transaction.DebugJTAJDBC` – transaction debugging. To get this level of tracing for Oracle, you need to use `ojdbc6_g.jar` instead of `ojdbc6.jar`.

- `weblogic.jdbc.rac.DebugJDBCONS` — low-level ONS debugging. A GridLink data source provides connectivity between WebLogic Server and an Oracle Database service, which may include multiple Oracle RAC clusters. It uses the Oracle Notification Service (ONS) to adaptively respond to state changes in an Oracle RAC instance, so with this sort of debugging you can identify problems with the ONS Client.
- `weblogic.jdbc.rac.DebugJDBCRCAC` — RAC debugging.

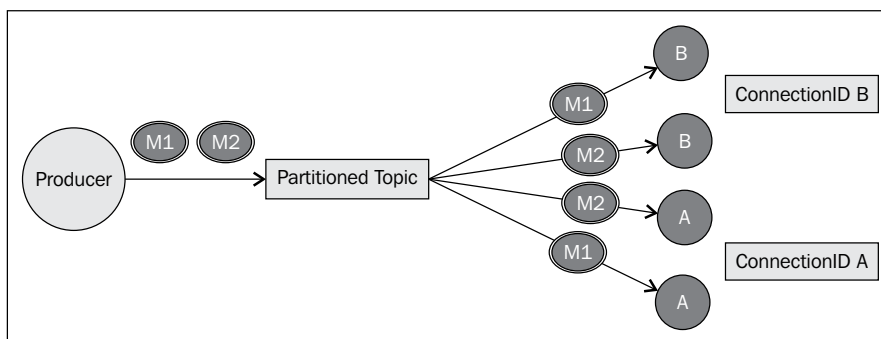
JMS Services

Java Messaging Services within WebLogic 12c hasn't been significantly changed. The version of JMS that is running is still at release 1.1.

The following are some of the changes:

- **Weighted Distributed Destinations** are deprecated since WebLogic Server 10.3.4.0. In a weighted distributed destination, the member destinations do not have a consistent configuration of all distributed destination parameters, particularly in regards to weighting, security, persistence, paging, and quotas. Recommended now is to use Uniform Distributed Destinations, because UDD can be used cluster wide.

Partitioned Distributed Topics give you the ability to load balance messages to members which provide a highly scalable and available publishing mechanism. This allows for more than one durable subscription to be made available for a particular subscriber.



WebLogic introduces the concept of an **unrestricted Client ID** policy. With this setting on a topic, more than one connection in a cluster can share the same Client ID. The standard option of restricted enforces that only a single connection with a particular Client ID can exist in a cluster. Unrestricted Client ID policy allows more than one JMS connection to use the same Client ID.

Shared Subscriptions allow multiple subscribers to share the same subscription which enables parallel processing of messages of a single subscription.

When creating a topic in WebLogic, set the **Forwarding Policy** to be partitioned. This causes a message sent to a partitioned distributed topic to be sent to a single physical member. In addition, the message will not be forwarded to other members of the cluster if there are no consumers in the current physical member.

If it's a high availability solution, then the listeners will need to connect to each and every physical member across the cluster. When a physical member becomes available/unavailable, WebLogic provides the `JmsDestinationAvailabilityHelper` API which listens to events relating to physical member availability and unavailability. Don't forget, the **Connection Factory** that is being used should have **Subscription Sharing Policy** set as **Shareable**.

- **New Message-Driven Bean (MDB) activation** configuration properties and deployment that provide high availability and parallel processing.

Security services

An important service to leverage these days, covering all kinds of malicious attacks, is security. In Java EE 6, there has been major improvements on various kinds of interfaces. WebLogic already provided end-to-end security for applications. WebLogic had already been proven of flexible in its security services and has a lot of plugins to many other vendor security software.

Java Authentication Service Provider Interface for Containers (JASPIC) support

JASPIC defines a service provider interface (SPI) from which authentication providers implement message authentication mechanisms. These mechanism can be used in web application's runtime containers

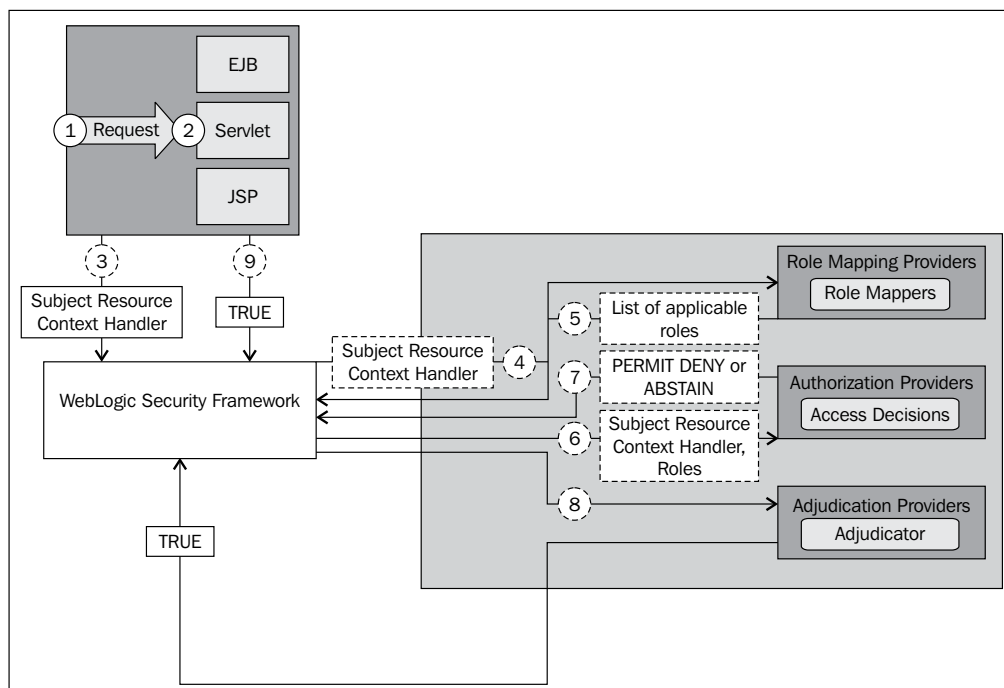
This message processing runtime uses this SPI to match and delegate the corresponding message security processing to the correct authentication providers configured in the security realm. The authentication provider returns a subject based on the authenticated user credentials Java Authorization Contract for Containers (JACC) 1.4 Support.

This specification works with the `javax.security.jacc` package, together with the `weblogic.security.jacc` package. The `weblogic.security.jacc` provides a `RoleMapper` interface, which allows the passing in of roles and principal names for JACC.

JACC is an alternative for the standard WebLogic Security Framework for EJBs and Web Applications. JACC extends the Java 2 permission-based security model to EJBs and Servlets, and is specified in JSR-115.

The WebLogic JACC provider fully complies with JSR-115. However, it does not support dynamic role mapping or authorization decisions for resources other than EJBs and Servlets. WebLogic JACC classes from the `javax.security.jacc` package are used for role-to-principal mapping, but for better performance, and for more flexibility regarding security features, it's better to use SSPI-based providers.

The following diagram shows you the security flow when using such a provider:



RSA JSSE Provider

A third-party provider, the **RSA JSSE Provider** is now supported in Weblogic 12c. It provides an interface for the Java Secure Socket Extension (JSSE). JSSE supports SSLv3 and TLSv1, plus some algorithms, and a list of Cipher suites for Transport Layer Security.

It can be statically registered in the JVM if you want to use it. In WebLogic 12c, you will have to enable it manually by editing `$JAVA_HOME/jre/lib/security/java.security` and set `com.rsa.jsse.JsseProvider`, as the first provider in the list.

```
#
# List of providers and their preference orders (see above):
#
security.provider.1=com.rsa.jsse.JsseProvider
security.provider.2=sun.security.provider.Sun
security.provider.3=sun.security.rsa.SunRsaSign
security.provider.4=com.sun.net.ssl.internal.ssl.Provider
security.provider.5=com.sun.crypto.provider.SunJCE
security.provider.6=sun.security.jgss.SunProvider
security.provider.7=com.sun.security.sasl.Provider
security.provider.8=org.jcp.xml.dsig.internal.dom.XMLDSigRI
security.provider.9=sun.security.smartcardio.SunPCSC
security.provider.10=sun.security.mscapi.SunMSCAPI
```

The JSSE standard API, available in the `javax.net` and `javax.net.ssl` packages, covers:

- Secure (SSL) sockets and server sockets.
- A non-blocking engine for producing and consuming streams of SSL/TLS data (SSLEngine).
- Factories for creating sockets, server sockets, SSL sockets, and SSL server sockets. With socket factories, you can encapsulate socket creation and configuration behavior.
- A class representing a secure socket context that acts as a factory for secure socket factories and engines.
- Key and trust manager interfaces (including X.509-specific key and trust managers), and factories that can be used for creating them.
- A class for secure HTTP URL connections (HTTPS).

SSL Implementation

SSL Implementation has been changed over several WebLogic versions, so let's have a look about what it will offer you in WebLogic 12c.

Because of the JSSE-based SSL Implementation, the Certicom-based SSL implementation is removed and is no longer supported. It supports stronger certificates, stronger than 128-bit such as TLS Cipher Suites or RSA.

Changes to SSLMBean

SSLMBean has been enhanced to support additional SSL configuration capabilities, including the ability to enable or disable the JSSE adapter.

WebLogic Server fully supports SSL communication, which enables secure communication between applications connected through the Web. WebLogic Server 12c includes support for using the Java Secure Socket Extension (JSSE) as the SSL stack for the following:

- Inbound SSL connections
- Outbound SSL connections that use the WebLogic SSL APIs (you can also call JSSE directly for outbound SSL connections)

As part of `ConfigurationMBean`, it indicates that the built-in SSL certificate validation should be used to complete and validate the peer's certificate chain, then the configured `CertPathValidator` security providers should be used to perform extra validation on the chain.

JSSE/SSL

JSSE is the standard SSL implementation in WebLogic Server 12.1.1. These items had to be changed to support it:

The default for `JSSEEnabled` has been changed to `true`. If it is set to `false`, it will be ignored. WebLogic overrides it and continues to use JSSE. It will only give you a warning, but the MBean will not be changed.

TLS 1.2 support

Together with SSL, WebLogic 12c supports the newest transport layer TLS 1.2. The TLS protocol provides communication security over the Internet. The protocol allows client/server applications to communicate in a way that is designed to prevent eaves-dropping, tampering, or message forgery. The primary goal of the TLS protocol is to provide privacy and data integrity between two communicating applications. The protocol is composed of two layers: the TLS Record Protocol and the TLS Handshake Protocol. At the lowest level, layered on top of some reliable transport protocol (for example, TCP [TCP]), is the TLS Record Protocol.

Usually the most recent version of the SSL or TLS protocol is good enough but clients may not support it. You can also specify the enabled SSL or TLS protocol based on circumstances (compatibility, SSL performance, and environments with maximum security requirements) that make the TLS V1 protocol more desirable for enabling acceptable SSL and TLS protocols.

Specifying the `weblogic.security.SSL.protocolVersion` system property in a command-line argument that starts WebLogic Server lets you specify the protocol that is used for SSL connections.

You can set startup command-line arguments so that WebLogic Server supports only SSL V3.0 or TLS connection:

- `-Dweblogic.security.SSL.protocolVersion=SSL3`
- `-Dweblogic.security.SSL.protocolVersion=TLS1`: This property value enables any protocol starting with TLS for messages that are sent and accepted such as TLS V1.0, TLS V1.1, and TLS V1.2.
- `-Dweblogic.security.SSL.protocolVersion=ALL`: This is the default.

Here you can see an example. Edit `startWebLogic.sh` and then add the following code:

```
export JAVA_OPTION="{JAVA_OPTIONS} -Xmx1024m -Xms1024m -Dweblogic.  
security.SSL.protocolVersion=SSL3"
```

Better support for Single Sign-On with Microsoft Clients

WebLogic 12c supports a security provider, the Negotiate Identity Assertion provider, to work with single sign-on (SSO) with Microsoft Active Directory clients. This identity assertion provider decodes Simple and Protected Negotiate (SPNEGO) tokens to obtain Kerberos tokens, validates the Kerberos tokens, and maps Kerberos tokens to WebLogic users. SPNEGO stands for Simple and Protected GSSAPI Negotiation Mechanism. It is a standardized interface for authentication (like JNDI is for directory look-ups) and the default implementation for SPNEGO under Windows is Kerberos (like LDAP is for JNDI).

WebLogic uses GSS-API to communicate with Kerberos. GSS-API is Generic Security Service API. It provides a common interface for accessing different security services, but most commonly Kerberos V5. WebLogic uses JAAS to authenticate to Kerberos. The JAAS framework and the Kerberos mechanism required by the Java GSS-API methods are built-in.

A **Negotiate Identity Assertion** provider is required in your WebLogic security realm in order to enable SSO with Microsoft clients. Furthermore, you need to use the following startup arguments when you start WebLogic Server for Kerberos authentication:

```
-Djavax.security.auth.useSubjectCredsOnly=false  
-Djava.security.auth.login.config=krb5Login.conf  
-Djava.security.krb5.realm=<ADRealm>  
-Djava.security.krb5.kdc=<ADhostname>
```

New features supported in this release are:

- Use of several new encryption algorithms for mapped user accounts that need to be encrypted. This version will support the default Windows RC4-HMAC encryption algorithm, and AES-128 and AES-256.
- Support for Java SE clients.

Web Services

In the new WebLogic Server 12c JAVA EE 6 plays an important role so, inevitably, Web Services also have new features and possibilities. Let's look at some of them.

WebLogic Web Services with Java EE 6

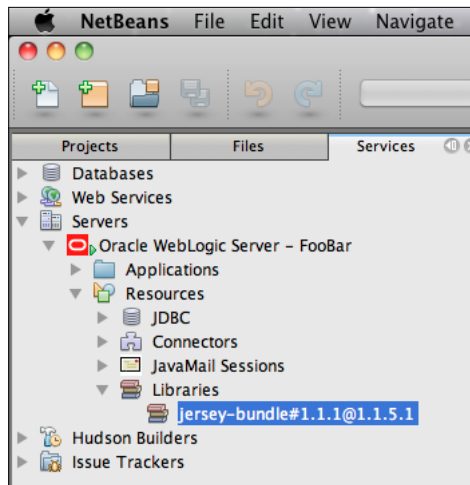
As been said before, WebLogic Server 12c fully supports the Java EE 6 specifications, so Web Services are also supporting these since 2009. But of course now, in 12c as well.

For instance, there is enhanced support for EJB 3.1. Support for Web Services in EJB 3.1 is based on the Java API for XML-based Web Services (JAX-WS) 2.1 specification, as well as its predecessor, the Java API for XML-based RPC (JAX-RPC) 1.1. If you want to manipulate the structure of your SOAP message, you can use an API called SAAJ (SOAP with Attachments API for Java). To register your XML, you can use JAXR.

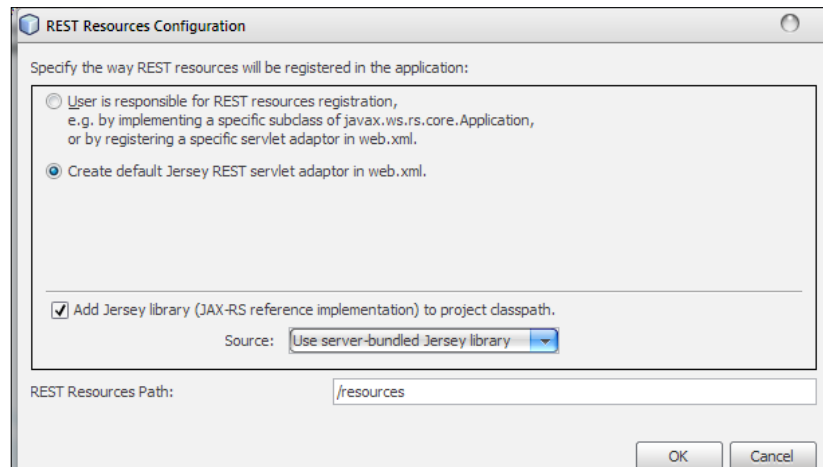
If you want to use some sort of remote protocol which is a bit similar to CORBA or RMI, the JAX-RPC interface is quite useful. Finally, they can be packaged together with an EJB in an EAR file.

WebLogic 12c and Jersey JAX-RS RI Version 1.9

Web Services in 12c now supports Jersey Java API for RESTful Web Services (JAX-RS) according to the JSR-311 specifications. WebLogic won't use shared libraries for this but uses Runtime MBeans for post-deployment manageability. Out of the box, the Jersey bundle is not yet implemented as a shared library in WebLogic, but after building and deploying to WebLogic, it shows up as a shared library; you can see the `jersey-bundle#1.1.1@1.1.5.1` library present on the server.



In the `weblogic.xml` deployment descriptor, you can see the references for JAX-RS shared-library that will be deployed on the server, which you specified during creating a building.



RESTful Services with Java (JAX-RS) support is provided through Jersey, which is the reference implementation for JAX-RS. This support was already added in 10.3.4 as part of the JAVA EE 6 specification. REST, other than SOAP, can be consumed by any client, even with Ajax and JavaScript. It is more lightweight because it does not parse XML, and consumes less bandwidth because it doesn't read the header every time.

Support for EclipseLink MOXy (JAXB)

EclipseLink JAXB (MOXy) is now the default JAXB (JSR-222) provider in WebLogic Server 12c. The EclipseLink MOXy component enables you to bind Java classes to XML schemas. For implementing mapping through annotations, MOXy uses JAXB, which also stores the mappings in an XML format. These mappings can be used for handling complex structures of XML code, but you don't have to mirror the XML schema into the JAVA class model anymore. The JAXB data binding process consists of the following tasks:

- **Bind** – Binds XML Schema to Java classes, or value classes. Each class provides access to the content via a set of JavaBean-style access methods. Binding is managed by the JAXB schema compiler.
- **Unmarshal** – Converts the XML document to Java objects that can be accessed by your Java code. Complex types and attribute declarations are mapped to field properties using class values which uses `get` and `set` methods.
- **Marshal** – Converts the Java objects back to XML content.

EclipseLink MOXy is one implementation of the standard runtime defined by the JAXB specification. To specify EclipseLink MOXy as your JAXB provider:

- Add the JAXB APIs and `eclipselink.jar` on your WebLogic Server's classpath
- Use a `jaxb.properties` file (in the same package as your domain classes

However, UDDI v2.0 Registry and UDDIExplorer, as well as WebLogic Web Services 8.1 Application Environment, are removed from WebLogic 12c.

Summary

Oracle WebLogic 12c has made a huge effort to make it compatible with a huge range of external systems, configurations, and so on. In almost every part, there have been major and sometimes minor updates to make the Java Application Server the most complete one.

Seeing all these new features, it is now time to look at the Cloud capabilities of WebLogic 12c. First, we will have a look at how we can manage and monitor WebLogic 12c with Enterprise Manager 12c.

5

Integration and Management with Enterprise Manager 12c Cloud Control

The more this book continues, the more we move on to the Cloud part which is part of Oracle's future strategy. Before we come up with Exalogic in the final chapter, we will first take a look at WebLogic's position in Oracle's Enterprise Cloud Management. In this chapter, we will see how we can scale up WebLogic 12 into Enterprise Manager 12c, how we can manage and monitor our environments, and more interesting features it can offer us.

Enterprise Manager was the first in the product-line which Oracle brought to 12c and it is the first complete Cloud Management Solution with Oracle Enterprise Manager.

Oracle Enterprise Manager 12c covers:

- Complete Lifecycle Management
- Integrated Cloud Stack Management
- Business-driven Application Management

What is Oracle Enterprise Manager 12c?

Oracle Enterprise Manager 12c (Cloud Control) is a complete solution for having a centralized monitoring system framework built with:

- An Oracle Database which hosts a repository
- WebLogic Server which runs a Java EE application
- An ADF GUI for doing administration tasks
- Client agents for sending notifications to Enterprise Manager 12c

Database hosts the so-called Oracle Management Repository(OMR), and WebLogic the Management Service (OMS).

The ADF Web GUI is called Oracle Enterprise Manager Cloud Control. And at last, there are server- and client-side plugins in the format of Oracle Virtualization. These plugins are for Oracle Virtual Manager based systems.

All together, these products are bundled into one common name: Oracle Enterprise Manager (OEM) or Enterprise Manager (EM).

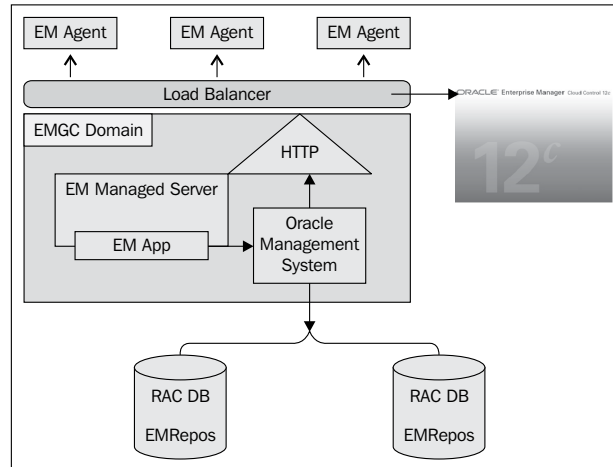
Oracle Enterprise Manager 12c system design

All the Oracle Enterprise Manager system components can be installed on one single host. For development or testing purposes, or learning environments, this is a good way to do, but in a production environment it is better to separate the various components on different hosts, for better performance and no single point of failure. Also for scaling purposes, this is a better way to do. For the Oracle Management Repository, you can use RAC for the Oracle Management Repository database.

When you want to scale out the WebLogic and Oracle Management Service tier, you can add a load-balancing solution to multiple frontend WebLogic servers hosting the Oracle Management Service. Adding a load balancer with additional WebLogic servers requires a virtual hostname for the WebLogic cluster, but in an existing Oracle Enterprise Manager environment, a reconfiguration of all of your Oracle Management Agents is necessary to resolve to the new virtual hostname. So when you deploy Oracle Enterprise Manager, consider using a virtual hostname for the web tier.

Enterprise Manager 12c (12.1.0.1) Grid Control requires a minimum of version WLS 10.3.5. This brings a new and improved interface, which also includes cloud computing management features such as charge back and metering.

In the following diagram, you can see an architectural overview of an Enterprise Manager system:



So a best practice, as you can see in this diagram, is to place the administrative web GUI and all the agents before the load balancer. They all can speak to one address in the form of a virtual hostname using the standard HTTP port 80 or even SSL for the GUI: and the agents connecting to the secure Management Port.

WebLogic Server Management: New in Enterprise Manager 12c

To integrate WebLogic Server 12c with Enterprise Manager 12c, there is a management pack available to provide a full and complete tool to manipulate, configure, monitor, and diagnose a WebLogic Domain. If you enable such a management pack, you are able to discover your WebLogic Server environments to integrate into your Enterprise Manager 12c.

The WebLogic Server Management Pack Enterprise Edition can be used for managing Oracle Fusion Middleware, Oracle WebLogic Server, and Oracle Application Server. This pack provides capabilities for application performance management, business transaction management, configuration management, service-level management, coherence management, as well as lifecycle management for Oracle Application Server, Oracle Fusion Middleware, and Oracle WebLogic Server software.

The features available with this pack are as follows:

- Configuration management features
- Application performance management features

- Service-level management features
- Coherence management features
- Business Transaction Management features
- Lifecycle management features

Configuration management features

The WebLogic Server Management pack provides you rich management features in order to configure, clone, and store your configurations. These configurations can include:

- **Detection of configuration changes:** You can create a historical image of a configuration, and some time later you can match a newly created configuration with this so-called golden image to detect if there are any changes and, if necessary, take actions on it.
- **Compliance and provisioning:** Provisioning is discussed later in this chapter. The compliance framework makes it capable to give insight if your systems match valid configurations, or if these systems are vulnerable to configuration changes. Also, the framework can advise you to solve these vulnerabilities. In fact, it is a defined set of rules to be used to the benefit of your systems. You can run reports periodically to let Enterprise Manager 12c advise you if and how to take actions.

WebLogic Server 12c provisioning and cloning

To be ready for a very demanding business, there is a possibility to clone your domain out of a provisioning profile.

You can create a provisioning profile entity with binaries and domain configuration or a middleware home entity with just binaries. You can clone a WebLogic domain or Middleware home from software library entities. There is a new out-of-box deployment procedure for deploying, redeploying, and undeploying Java EE applications from the Cloud Control Console. You can now access provisioning operations from the WebLogic domain menu.

You can clone WebLogic Domain from:

- **Reference Install:** This reduces time and eliminates errors in building environments. The clone operation includes WebLogic Server binaries and domain configuration files.

- **Software Library:** Creates components in Software Library for Middleware Home binaries and/or WebLogic Domain configuration. You can clone such components to new hardware and specify domain configuration such as listen addresses, ports, data sources, JMS stores, and security store/providers.

Provisioning

Source Destinations **Domain Configuration** Schedule Review

Middleware Provisioning : Domain Configuration

Domain Properties

Specify properties for the domain. The username specified is the default administrator, and the Unique Domain Identifier is used as a prefix to ensure farm names are unique in environments with the same domain name.

* Domain Name base64_domainCloned1Cloned

* Domain Administrator Username base_user

* Password [masked]

* Confirm Password [masked]

Description

* Unique Domain Identifier Farm01

* Domain Location /scratch

Back Step 3 of 5 Next Cancel

Cloning and provisioning can be achieved by using the **Middleware Provisioning** tool.

By using the Middleware Provisioning deployment procedures you can:

- **Clone a WebLogic Domain from an existing installation:** You can clone a WebLogic Domain from the existing installation. The **Clone WebLogic Domain** option starts a wizard for cloning a WebLogic Domain from an already existing reference domain that is already discovered or registered with Cloud Control. In this cloning process you will have to specify the following:
 - **Host Credentials**
 - On the Domain Properties page:
 - Domain Name, Domain Location** (on file system)
 - Domain Administrator Username and Password**
 - Unique Domain Name Identifier** – used to name the farm target the same as the WebLogic domain name but with the `Farm_` prefix
 - Node Manager port address
 - Other resources such as JDBC, JMS, Security, Logging
- Keep in mind to use a unique port for the AdminServer as it might conflict with other already existing configurations.

The following screenshot shows you the cloning wizard:

The screenshot shows the 'Provisioning' console with the 'Domain Configuration' step selected. The left pane shows a tree view with 'WebLogic Domain' expanded, showing 'Basic Configuration' and 'Advanced Configuration'. The main pane is titled 'Middleware Provisioning : Domain Configuration' and shows the 'Administration Server' configuration. The 'Name' is 'AdminServer', the 'Host' is 'wlspackt.com', the 'Listen Address' is 'wlspackt.com', the 'Listen Port' is '17001', and the 'SSL Listen Port' is empty. The 'Enable SSL' checkbox is unchecked. The 'Back' button is disabled, and the 'Next' button is enabled.

- **Cloning a Middleware Home from an existing installation:** To do this, keep in mind that the hosts on which the Middleware domains are to be cloned must be registered in Cloud Control. Also, read permissions are required on the Middleware Home directory on the host machine on which the Administration Server is running.

The following screenshot shows you the Middleware Cloning Wizard:

The screenshot shows the 'Provisioning' console with the 'Source' step selected. The left pane shows a tree view with 'WebLogic Domain' expanded, showing 'Basic Configuration' and 'Advanced Configuration'. The main pane is titled 'Middleware Provisioning : Source' and shows the 'Select source from an existing installation' step. The 'Host' is 'wlspackt.com'. The 'Source Information' table shows the following data:

Component	Location
Middleware Home	/middleware_home
WebLogic Server	/middleware_home/wlserver_12.1

The 'Host Credentials' section shows the 'Preferred' radio button selected. The 'Preferred Credential Name' is 'Normal Host Credentials'. The 'Credential Details' table shows the following data:

Attribute	Value
UserName	.
Password	*****

The 'Back' button is disabled, and the 'Next' button is enabled.

- **Cloning from a WebLogic Domain Provisioning Profile:** With this you can clone a WebLogic Domain from an already existing profile present in the software library of the Enterprise Manager 12c.
- **Cloning from an Oracle Middleware Home Gold Image:** If you have installed WebLogic software on a host and you want to keep an image of it, you can create a so called Oracle Middleware Home Gold Image and save it in the Software Library. Later, you can then use this gold image as the source for future Middleware Home installations.

You will have to calculate how much an image (gold or domain provisioning) will consume disk space in the software library. A rule of thumb for Golder Images and Domain Provisioning is as follows:

- For Golden Images: Middleware Home Size + Space for temporary scripts
- For Domain Provisioning: Middleware Home Size + Domain Home Size + Space for temporary scripts

Automating discovery and detecting configuration changes

Oracle Enterprise Manager not only gathers a lot of configuration information about WebLogic Server, but also its underlying hardware and operating system.

Templates for specifying what configuration items should be collected for Oracle WebLogic Server and all its related components are available out of the box and can be customized to collect only the relevant configuration items that IT personnel require. Examples of information collected at regular intervals include:

- WebLogic Server software installations, including applied patches
- WebLogic Server configuration parameters (for example, ports, JVM information, JDBC JMS resources, startup, and shutdown classes) and configuration files
- Operating system settings, patches, kernel parameter settings, and installed packages
- Hardware components such as CPU, RAM, disk storage, and network devices

Configuration changes can be also monitored across the entire WebLogic Server stack. From the application down to the hardware it is possible to detect all changes for a specific configuration between a specified time frame. Changes that are applied to an environment that previously worked fine but is suddenly not performing at an acceptable level can be discovered very quickly.

The following screenshot shows you the use of a typical WebLogic Comparison Template:

Edit Comparison Template

Target: Oracle WebLogic Server

Type:

* Template Name: My WLS Config Compare Template

Template Owner:

Description:

Template Settings

- Configuration Files (Data Contents)
- Target Properties
- Application
- Node Manager Configuration
- Machine Configuration
- Web Service Configuration
- Web Service Port Configuration
- JRF Web Service Configuration
- JRF Web Service Port Configuration
- JRF Web Service Operations Config
- JRF Web Service Policy References
- Resource Adapter
- Resource Adapter Outbound
- Web Modules
- EJB Modules
- Server Information**
- JDBC Connection Pool
- JDBC Datasource

Property Settings

Property Name	Ignore Differences	Notify on Differences
Java Version	<input type="checkbox"/>	<input checked="" type="checkbox"/>
Listen Port	<input type="checkbox"/>	<input type="checkbox"/>
Login Timeout	<input type="checkbox"/>	<input type="checkbox"/>
Low Memory GCThreshold (Percent)	<input type="checkbox"/>	<input type="checkbox"/>
Low Memory Granularity Level (Percent)	<input type="checkbox"/>	<input type="checkbox"/>
Low Memory Sample Size	<input type="checkbox"/>	<input type="checkbox"/>
Low Memory Time Interval	<input type="checkbox"/>	<input type="checkbox"/>
Machine Name	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Maximum Open Socket Count	<input type="checkbox"/>	<input type="checkbox"/>

WebLogic Server 12c monitoring

Monitoring your WebLogic Server environment can be done in many ways, through scripting (WLST) or third-party vendors, but with the Enterprise Manager 12c you will get a rich set of possibilities to watch and monitor how your system is doing it now, in the past, and even in the future.

Performance monitoring and diagnostics of WebLogic Server

Enterprise Manager 12c provides a wide variety of monitoring and diagnostics options for WebLogic and its entire Middleware environment.

These monitoring options are very rich and every administrator can choose his/her own favorites. For a typical WebLogic domain, you would like to monitor items such as:

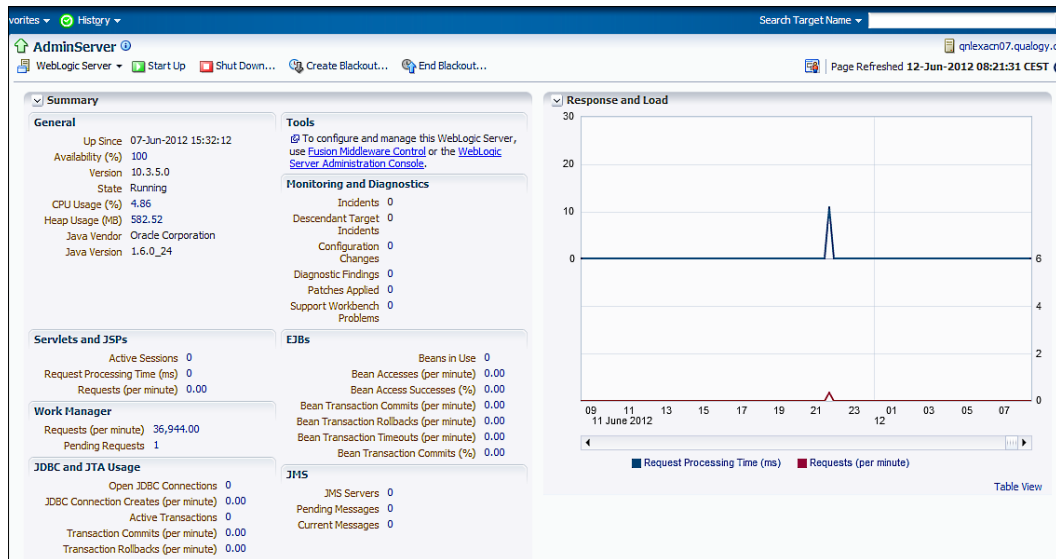
- Clusters and servers
- Applications (servlets, JSPs, EJBs)

- Resources (JDBC connection pool, data sources)

Predefined metrics:

- Performance and availability
- Real-time monitoring
- Historical monitoring for trending and reporting

The following screenshot shows you the WebLogic Server summary page:



The following capabilities are available:

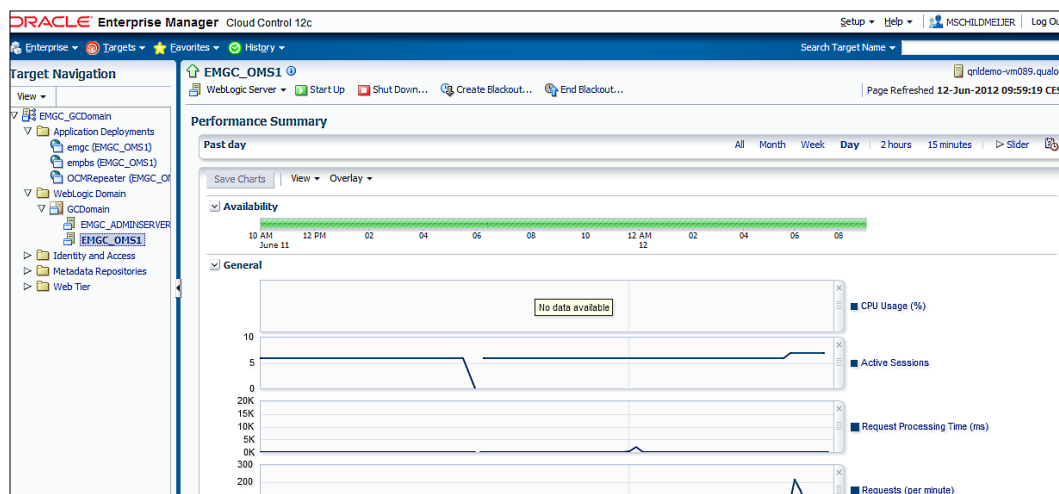
- Customizable performance summaries
- Out-of-box metrics
- Metric extensions
- Composite application dashboard
- Request monitoring
- JVM diagnostic
- Middleware diagnostic advisor
- Diagnostic snapshots
- Monitoring for deployed applications
- Application components dependency and performance

- Log viewer
- Event monitoring

Let's discuss the previously mentioned capabilities in detail.

Customizable performance summaries

With customizable performance summaries one can analyze and correlate performance data more efficiently. This goal can be obtained by specifying time range from where to display data. You can select a time range to watch and analyze trends in your WebLogic Domain.



You can choose charts to be displayed, arrange order of charts, and display data from multiple components in single chart. For a performance trend analysis, you can use the option of saving a baseline of current performance data to be compared with future data possible.

Out-of-box metrics

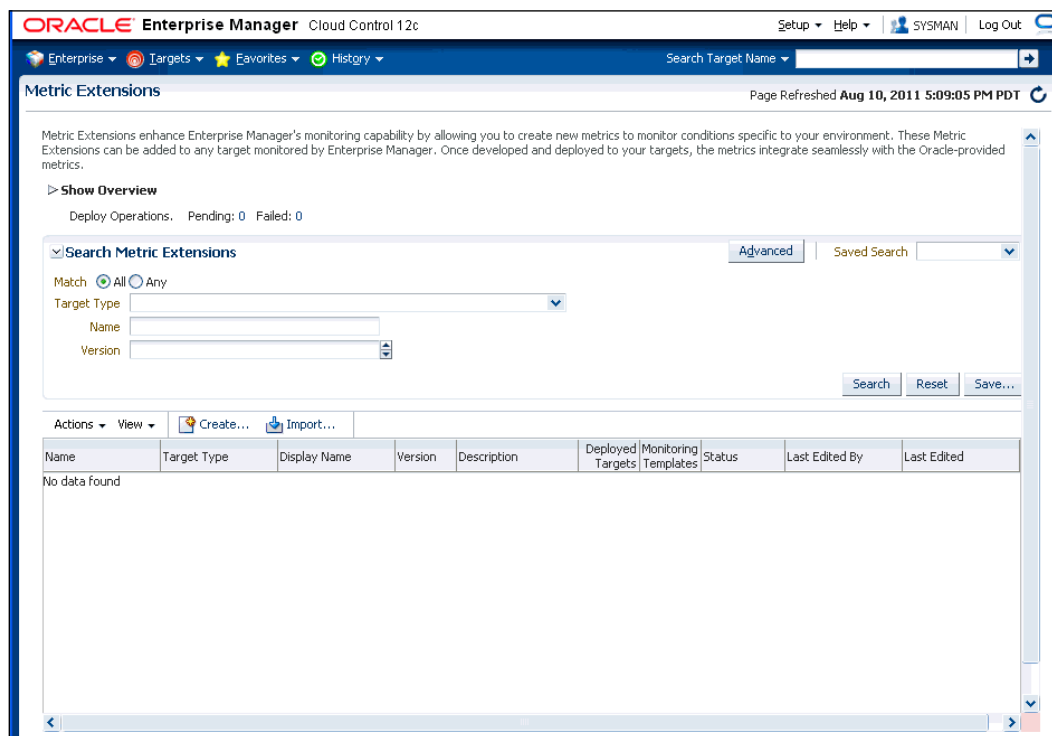
Default Enterprise Manager has some out-of-the-box performance metrics for each Middleware target. Metrics which tell you the information about WebLogic domains, clusters, applications, Web Services, and many more. All the data about all these different components end up being stored in the database Management Repository.

Some examples that Enterprise Manager can automatically monitor are:

- CPU and memory of WebLogic Server, including detailed monitoring of individual JVMs
- Response times of Java applications by pinning into Enterprise Java beans and Servlets
- Oracle HTTP Server statistics like error rates, connection times and durations, and response times
- A list of top ten servlets based on the number of requests that have been fired to these servlets, about how long and what average time they process their requests

Metric Extensions

With **Metric Extensions** you can create metrics on any target type and customize these metric thresholds and collections. **Metric Extensions** can create metrics for a lot of target types.

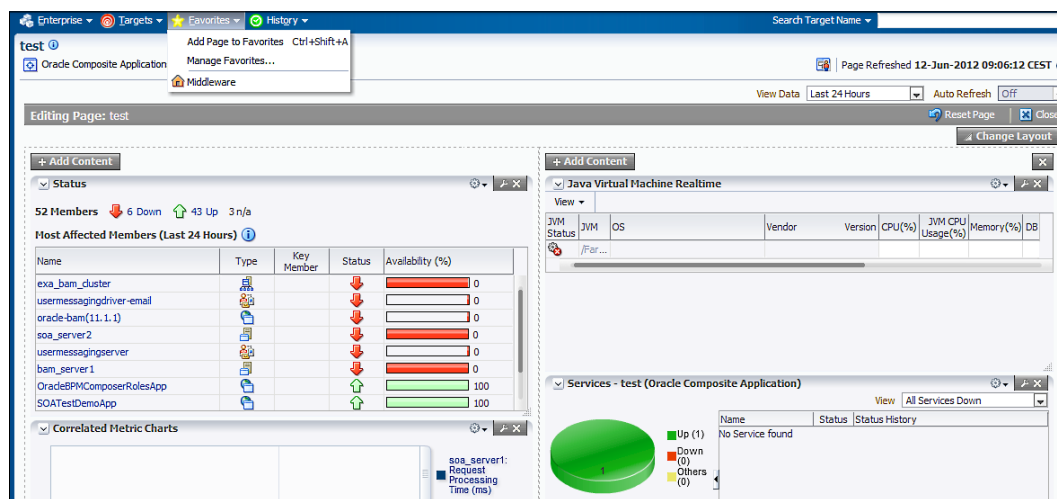


Composite Application dashboard

The **Composite Application** dashboard gives full visibility into both service-level metrics as well as critical component-level metrics across any composite application. As composite applications consist of both Java EE and SOA components and perhaps other key middleware technologies such as Oracle Coherence and/or Oracle Service Bus, it is critical to provide a single dashboard view across the application with key indicators of the application health as well as some quick diagnostics to identify problems in an early stage.

Service tests can be set up to facilitate proactive monitoring of the composite application with service levels tracking the health of those service tests. JVM, WebLogic Server, Application Deployments, and Host metrics are available together with an incident console tracking all of the alerts or policy violations that might occur on those tiers.

Finally, the dashboard can be customized however you see fit to include any metrics and as many monitoring or diagnostics regions that are required, with specialized regions that can be added to the dashboard for many monitored target types.



Request Monitoring

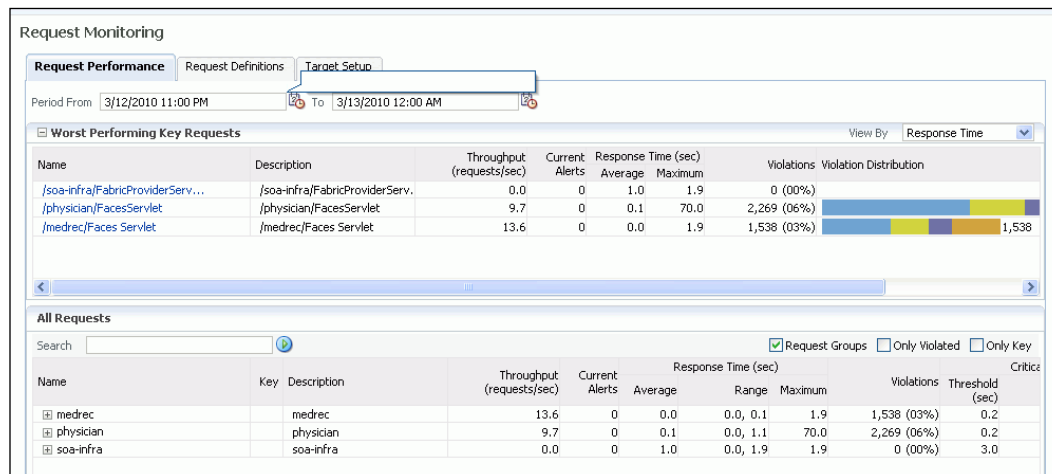
Request Monitoring gives an end-to-end visibility into requests and helps localize end-user performance problems based on the application deployment model. In a way, you can visualize how servers interact with each other to deliver business end-user services requests. There is a possibility to trace end-user requests from the client to endpoint across all the servers and applications associated with each transaction.

Only synchronous transactions can be monitored that are running on WebLogic servers.

Some features of **Request Monitoring** are:

- You can do a transaction capture and trace calls of these transactions
- You can diagnose requests performing badly by detecting problematic servers with bad service time
- Faster fault discovery, and you can specify a level of request and response time to which your components should comply to
- You can use the JVM Diagnostic feature to do diagnostics about performance and so identify fault reasons

The following screenshot shows you the **Request Monitoring** summary page:



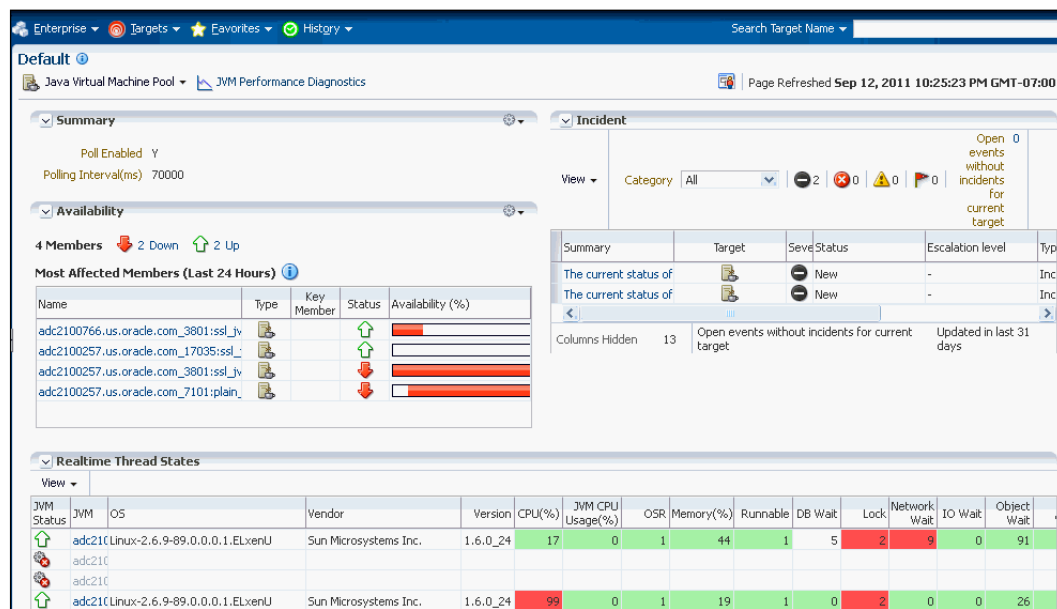
JVM Diagnostics

JVM Diagnostics in Enterprise Manager 12c can be used as a JVM diagnosis tool which has minimal impact on the JVM. You have real-time and historical monitoring and diagnostics always on. It occurs very often that Java applications often have availability and performance problems. A lot of time is spent diagnosing the root cause of these problems. Many times, the problems occurring in production environments either cannot be reproduced or may take too long to reproduce in other environments. Oracle Enterprise Manager Cloud Control 12c's JVM Diagnostics enables administrators to diagnose performance problems in applications in a production environment. You do not have to reproduce problems, which improves application availability and performance.

Using JVM Diagnostics, administrators will detect in a quick way the root cause of performance problems without replaying them in the test or development environment. It does not require complex instrumentation or restarting of the application to get in-depth application details. It is even possible to drill down from Java problems to database issues that are causing application downtime without any detailed application knowledge.

You don't need any application instrumentation or any server restarts. It gives you a complete visibility into the JVM stack heap and threads. You can analyze the impact bi-directionally: JVM to a database or vice versa.

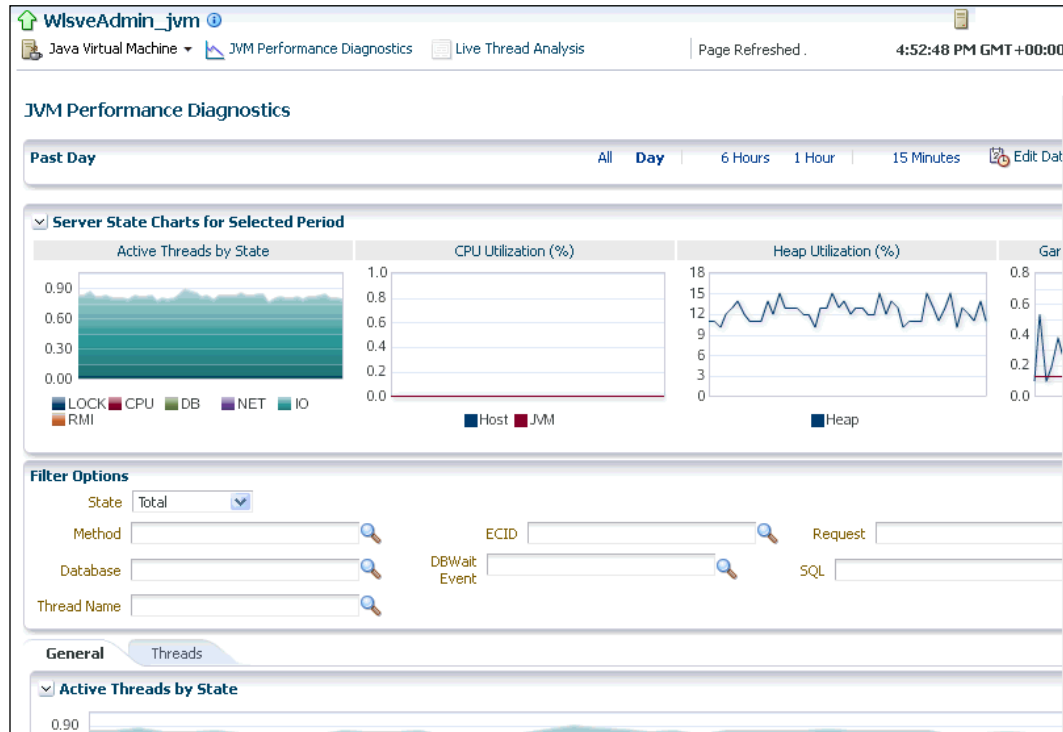
JVM Diagnostics can be deployed on any JVM (that is, Sun, JRockit, and IBM).



You can monitor a specific JVM in a pool, view historical and real-time data, and so on. You can do a lot of diagnosis by accessing the JVM home page and:

- View JVM Performance Diagnostics
- View JVM Performance Summary
- View the Live Thread Analysis Page
- View the Live Heap Analysis Page
- Manage Thread Snapshots
- Manage Heap Snapshots
- Setup JVM Properties

The following screenshot shows a typical JVM home page:



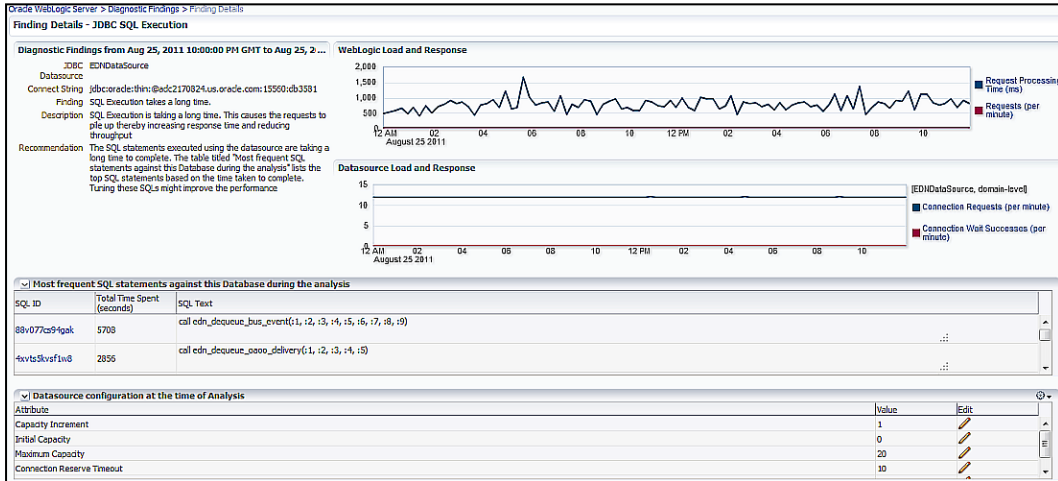
Middleware Diagnostics Advisor

The Middleware Diagnostics Advisor analyzes the entire WebLogic stack and shows diagnostic findings and tries to get behind the root cause of a problem. It correlates and analyzes the input and offers advice on how to resolve the problem. For example, it can help you identify that slow SQL statements or a JDBC connection pool are causing a performance bottleneck.

You can view the diagnostic findings for one or more servers in a WebLogic Domain if the Middleware Diagnostics Advisor has been enabled.

You can create a diagnostic snapshot which provides a collection of both JVM and WebLogic Server diagnostics and log data that can be exported or imported into other Cloud Control systems for analysis at some other point in time.

The following screenshot shows SQL execution diagnostic finding with detailed analysis of slow SQL in your application with links to further analyze and tune the SQL:







Diagnostic Snapshots

When a WebLogic system reaches a critical state, the first thing is to solve the problem(s) that occur, but you probably would like to quickly capture diagnostics in a production environment in order to later analyze in the future.

This feature within Oracle Enterprise Manager gives you the ability to take a snapshot that will correlate both the JVM state (threads, garbage collection, RAM, CPU) and the overall WebLogic logs across one or more servers. They can also be exported to either another Oracle Enterprise Manager Cloud Control or directly to Oracle support. This ensures that the diagnostics snapshot is later available for analysis in order to find the root cause and create a permanent fix to the problem.

The following screenshot is an example of the Diagnostics Snapshots Page of a JVM:

wsve_domain_jvmpool 

Java Virtual Machine Pool  JVM Performance Diagnostics  Page Refreshed Jul 31, 2011 5:57:19 PM GMT+00:00 

Upload Thread Snapshots

You can upload a Thread Snapshot from the client machine to the OMS. To upload a Thread Snapshot to a particular location on the OMS, Edit the Thread Snapshot path which represents the location on the OMS where Thread Snapshots will be uploaded

Upload Diagnostic Image

Thread Snapshot Path [Modify](#)

Thread Snapshot File [Browse...](#)











[Start Upload](#)

Locate Thread Snapshots and Load Them into Database

This table shows the list of Thread Snapshots present at the location specified by the value of Thread Snapshot Path. You can edit the Thread Snapshot Path to select a different Thread Snapshot root directory on the OMS

Thread Snapshots Located On Server

Thread Snapshot Path [Modify](#)

Filename	Date	Size (bytes)	Load	Delete
traceactive.1311931827171.zip	July 29, 2011 9:31:03 AM GMT	1361		
traceactive.1311774174690.zip	July 27, 2011 1:43:05 PM GMT	1463		
traceactive.1311932010212.zip	July 29, 2011 9:34:03 AM GMT	1309		
traceactive.1312133538017.zip	July 31, 2011 5:32:52 PM GMT	1329		
traceactive.1311934643613.zip	July 29, 2011 10:17:33 AM GMT	1410		

Loaded Successfully, Goto to saved Thread Snapshot

Monitoring for deployed applications

Enterprise Manager can integrate application instrumentation in the Enterprise Manager Event monitoring infrastructure. In an application, a developer can build some sort of JMX or Web Service operation, so for this you can build your own management plugin using command-line tools.

You can add performance metrics for JMX-instrumented applications deployed on Oracle WebLogic Server. To add, use a command-line tool `emjmxcli` to automate the generation of the target metadata and collection files. All JMX-enabled applications deployed to the WebLogic Server can be consolidated and monitored.

With `emjmxcli` you are able to monitor JMX Applications deployed on WebLogic Server. Monitoring JMX-instrumented applications with Enterprise Manager entails defining a new target type that Enterprise Manager can monitor via Management Plugins.

The JMX command-line tool (`emjmxcli`) simplifies creating the requisite target definition files—metadata and the default collection file.

The tool is an offline configuration utility that connects you to MBeanServer and enables you to browse available MBeans.

To start the JMX command-line tool:

1. Go to the `$AGENT_HOME/bin` directory.
2. Run any of the following commands:

```
emjmxcli -t WebLogic [OPTIONS]
emjmxcli -t JVM [OPTIONS]
```

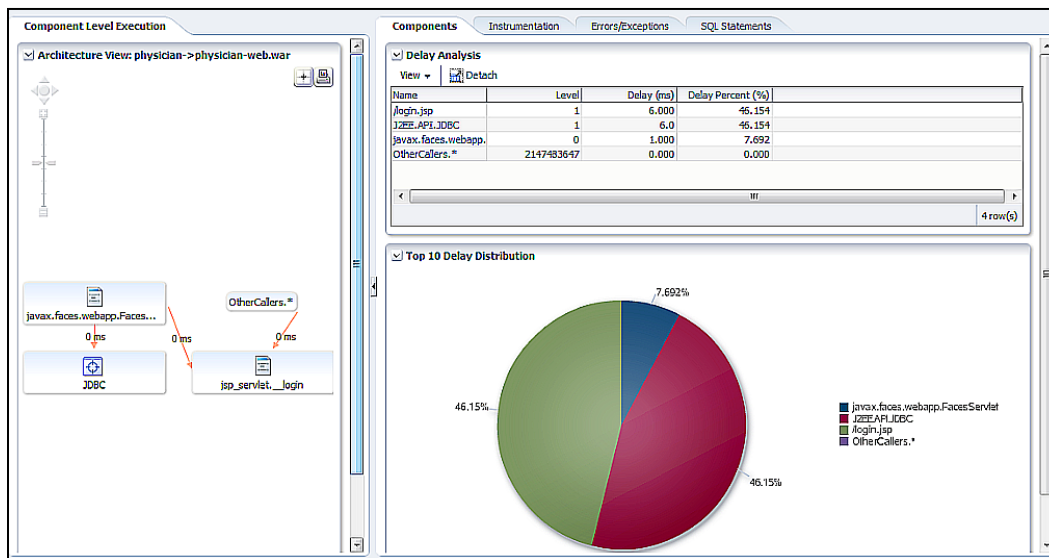
Once invoked, the command-line interface automatically prompts you for the requisite information

Application components dependency and performance

Java EE metadata can be complex and abstract and this complexity keeps growing with the introduction of new frameworks. The difficulties lie between the various application components such as Servlets, JSPs, ADF, EJBs, and the corresponding code. A Middleware Administrator should understand the metadata defining those relationships.

With Oracle Enterprise Manager 12c, administrators can view dependencies and relationships between high-level components such as JSPs, Servlets, Portlets, and Web Services and the underlying Java EE components that support those services such as EJBs and JDBC calls.

The following screenshot shows invocation count for a servlet and its underlying components giving an idea of the flow of context for that URI. Also, administrators can check how much time is spent in each of the components via the delay analysis metrics provided in the associated pie chart and table. Administrators can then drill down deeper into each class or component to find out how it behaves based on the context from which it was originally called. This is particularly useful considering that many components in Java EE applications are considered shared components where context is critical.



Log Viewer

With the Log Viewer, you can gain access to log files regardless of where they reside. You can access WebLogic and Fusion Middleware log files from a single console, search and correlate messages across log files based on time, severity, or Execution Context ID (ECID).

Also, it is possible to download log files or export messages to a text, XML, or CSV format.

If you like to control WebLogic Server Log Files, it is possible to use Oracle Management Service, a Java EE application deployed on an Oracle WebLogic Server.

You will find the log file information of the WebLogic Server components at:

```
<EM_INSTANCE_BASE>/user_projects/domains/<domain_name>/servers/<SERVER_NAME>/logs/<SERVER_NAME>.log
```

You can specify rotation by size or time, as well as the number of files to keep before it will be deleted. The default settings are:

- In production mode, rotation up to 5 MB
- Level of log messages is set to **WARNING**
- The maximum number of files is 10

Event monitoring

With event monitoring, one can be proactive about availability and performance problems 24 x 7. You can specify critical versus warning thresholds for metrics.

There are various notification methods such as e-mail/page, SNMP trap, or OS command. Also, notification rules and schedules can be created for when to receive alerts.

Business Transaction Management

End users definitely like to know how their business transactions are doing, so Enterprise Manager provides Business Transaction Management. These end-to-end transactions are then monitored in real time and you can extract business KPIs from the payload. Transactions can be searched for and aggregated to better trace, track, and troubleshoot problems. They can be discovered because the transaction contains some sort of tracking algorithm.

This BTM feature is available for Java EE applications and web applications and provides the following information:

- **Exception Management:** When an exception or error occurs in any transaction, it will locate exceptions and errors in transactions across multiple application components.
- **Transaction discovery:** Transaction flows can be discovered, recorded, and correlated about applications across multiple WebLogic Server Instances.
- **Transaction Level Agreement:** You specify your transactions to meet a certain kind of level agreement. With this feature you can easily detect if a transaction does not meet this level.
- **Contextual visibility:** You can entirely drill down your Java application to see the contextual visibility and relate them metric, to understand and analyze potential application bottlenecks and performance trends. A result of these analysis is you can issue capacity changes and perform overall application management.

Heap Analysis

Your Java Virtual Machines, in which your applications are running their code, should be monitored if it does not contain memory leaks. To find these memory leaks, you can use Enterprise Manager 12c to take and analyze snapshots of the JVM heap. You can do a live Heap Analysis or perform a snapshot to analyze the garbage collector of the JVM. You can also compare a snapshot with earlier taken snapshots, and isolate object that consume a lot of memory.

Integrated Cloud Stack Management

To service the entire Oracle Cloud Stack, Enterprise Manager 12c provides manageability through your entire Cloud environment. All manageable capabilities are represented to manage Exalogic and Exadata, and Sun Hardware through Oracle Ops Center 12c. Oracle Ops Center is a tool to control and administer your hardware such as CPU, RAM, Network Interfaces, and Storage.

You can do full performance monitoring for E-business Suite, Siebel, PeopleSoft, JD Edwards, and Fusion Applications. Also, you can perform user experience management, change and configuration management, patching, provisioning, testing, performance management, integrated diagnostics, and automatic tuning.

Quality management has a complete test suite for functional and load testing for testing web applications, doing a replay of a test scenario and real live testing, and simulating production scenarios.

So in general, Integrated Cloud Stack Management makes it easier for administrators to control their environment with the help of these tools.

Summary

In this chapter, we tried to provide you a brief overview about how Oracle WebLogic Server 12c is integrated into Oracle Enterprise Manager 12c. A WebLogic administrator can extend the capabilities of the WebLogic Server environments that are controlled by the Enterprise Manager.

In the next chapter, we will focus more on the cloud with WebLogic as key application server in Oracle's engineered systems.

6

Oracle WebLogic 12c to the Cloud: Exalogic

Oracle WebLogic Server 12c is the latest release of the application server for conventional systems, engineered systems, and cloud environments. As the center piece of Oracle's Cloud Application Foundation and a core part of the Oracle Fusion Middleware product family, Oracle WebLogic Server continues to deliver innovative new capabilities for building, deploying, and running **Java Platform Enterprise Edition (Java EE)** applications.

Oracle WebLogic Server 12c is focused on developer-centric and cloud-centric issues through enhancements and advanced technology integrations that include its distributed data grid as well as deployment and management improvements. WebLogic 12c server is the beginning of a next-generation foundational middleware platform that automates and eases deployment and management functionality.

WebLogic Server is still a work in progress (for example, 12c needs to be integrated with Oracle Fusion Middleware). However, this kind of support for standards-based development and integration with its engineered systems will help drive interest in its private cloud platform, Exalogic, and be ready for middleware, applications, and database technologies to be supported in its private cloud offering.

Let's see Oracle WebLogic 12c's position in Oracle's engineered system.

What is Oracle Exalogic?

Exalogic is a computer appliance made by Oracle, or better called an engineered system where software and hardware have been put into one big box. First of all, let's have a look at the hardware: Oracle Exalogic is a rack of up to 30 compute nodes mounted in a rack with Infiniband backplane and ZFS storage (ZFS is a combined file system and logical volume manager designed by Sun), where each compute node has two Intel Xeon x86 CPUs with six cores each. This means that a full rack has a total of 360 cores. All of these individual servers are interconnected with each other via Infiniband networking with the ability to connect together up to eight racks of Exalogic or Exadata on one Infiniband network. Each of these 30 servers has 96 GB of RAM and 64 GB of Solid State Disks. Additionally, the rack has 60 TB of SAS disk storage to be shared between those 30 servers. All this sounds fairly impressive on paper, however unlike traditional hardware, customers don't have the flexibility to select the components they need in this hardware configuration, except to buy a quarter of the rack, a half rack, or a full rack.

Inside an Exalogic Machine you will find:

- **Integrated Storage:** Each Exalogic hardware configuration includes an Infiniband-attached ZFS storage cluster usable for application binaries, log files, or any other application requiring high performance, highly available disk storage. The storage is shared for applications, and clustered. It consists of 40 TB disks with 4 TB read cache and 72 GB write cache.
- **Infiniband I/O Fabric and 10 GbE:** Infiniband is a high speed connection bus for internal and external connections in Oracle's engineered systems such as Exadata and Exalogic. All software components within Oracle Exalogic are optimized to use this high-speed Infiniband bus, which can perform up to a speed of 40 GB/s in theory, of course. Connections between an Exalogic and Exadata can go over this Infiniband bus, but its limit is when it goes outside the box, because there is no Infiniband interface available. However, this can be emulated using Ethernet over Infiniband (EoIB) which can perform up to 10 GB/s. As you can see at the specifications, Infiniband performs much faster than the traditional hardware used in conventional systems.

The WebLogic Exalogic optimization was introduced from version 10.3.4, which you could enable in the WebLogic Admin Console.

The following screenshot shows you where to enable this optimization:

The screenshot shows the WebLogic Configuration console. The 'Configuration' tab is selected, and the 'General' sub-tab is active. The domain name is 'WLSTest'. The 'Enable Administration Port' checkbox is checked. The 'Administration Port' is set to 9002. The 'Production Mode' checkbox is checked. The 'Enable Exalogic Optimizations' checkbox is checked and highlighted with a red box.

This Exalogic Optimization is represented by Exalogic Optimizations Enabled MBean. The optimization is needed because it will have to process the immense performance boost an Exalogic system delivers. Also, WebLogic thread management and request processing are brought up to speed to serve the Java applications, which are running in this high-speed environment. To enable this in the console, go to the **Domain Structure** pane and click on the domain name.

- **Compute Nodes:** The Compute nodes in the Exalogic system can be up to (30x) Intel Xeon x86 compute nodes with redundant Infiniband connectivity, power and solid state disks. Its CPUs can be up to 360 Xeon cores (2.93 GHz), 12 cores per server, 8/16/30 server configurations (per rack). Memory in a full rack is 2.8 TB DRAM, 960 GB SSD.

Exabus

The technique that connects all system components with each other is called Exabus. It is an I/O-based communication fabric to provide the basis for Exalogic's reliability, scalability, and performance.

It has the function of extending and connecting the PCI e-based system bus used within each of the major system components. Exabus is based on Quad Data Rate (QDR) Infiniband. QDR is a technique that takes advantage of the multicore CPU architecture which enhances CPU speed and consists of hardware, software, and firmware distributed throughout the system and involving every major system component.

The Infiniband specification was developed by Compaq, IBM, and Hewlett-Packard, with Next Generation I/O developed by Intel, Microsoft, and Sun Microsystems. This technique has been used by Oracle for its own engineered system, which is in fact a high performance computing system. It provides the greatest available bandwidth per physical port (40 GB/s – in theory) and lowest latency (~1.07µsec).

Since the traditional IPv4 has limitations, IPv6 is supported to be able to expand your devices.

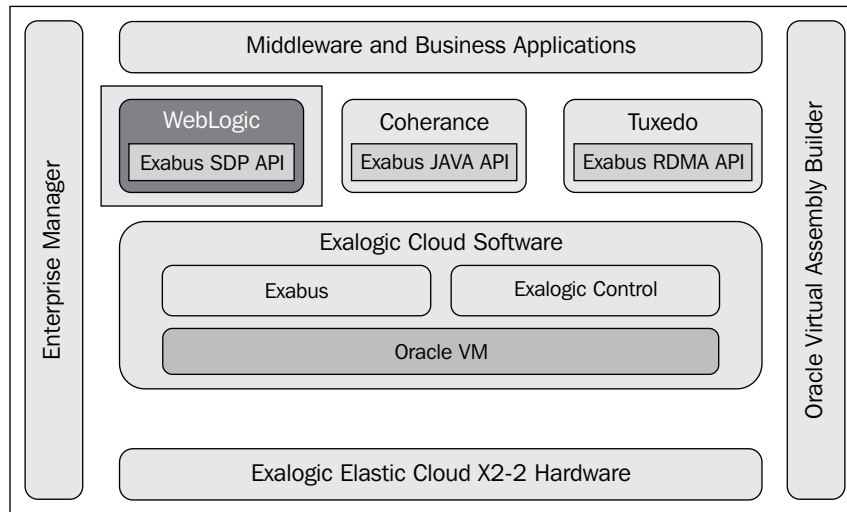
Other protocols supported are IP (IPoIB) and Ethernet (EoIB). In this case, already existing applications do not need to be modified but can use the benefits of this new interface performance.

Infiniband communicates directly to applications, and can isolate these applications using channels for efficient communication.

Oracle Exalogic Cloud Software components

The Oracle Exalogic Cloud Software components are used to increase the performance of deployed applications. Exalogic can improve performance of SOA or other message-distributed applications that make extensive use of enterprise messaging. It can handle large volumes of HTTP requests. Applications that have large memory requirements or that are highly multithreaded are no problem for Exalogic. It can also execute large volumes of transactional interactions with Oracle 11g Database and/or RAC instances, by executing a lot of actions in memory.

The following diagram shows how various main components and WebLogic 12c are integrated in the Exalogic stack:



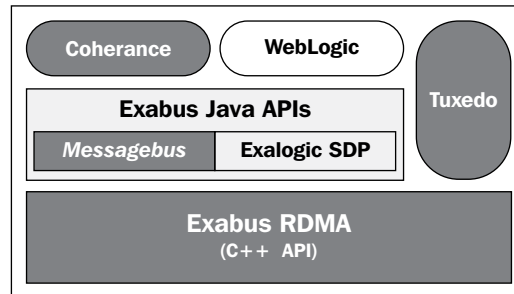
Exalogic Cloud Software

Exabus uses low-level drivers which operate directly on kernel level, **Remote Direct Memory Access (RDMA)** which loads directly into memory instead of acting directly on OS level. Kernel-level Java Virtual Machines are optimized using memory buffers which leads to fewer garbage collects, and this of course, leads to higher performance, because garbage collection can be an intensive action to perform. Having more memory available, the need of doing frequent garbage collects disappears and can be done at an appropriate time so the application does not notice anything of these actions.

The Exalogic Software consists of three APIs:

- RDMA API is the API being used to accelerate Tuxedo and make use of all benefits of all the extreme performing hardware
- The Message bus API is meant for accelerating Coherence
- Sockets Direct Protocol (SDP) API is meant to accelerate WebLogic

In the following diagram, you can see the position of the Exabus stack and, in particular, Tuxedo:



RDMA API: Oracle Tuxedo

Oracle Tuxedo is an open system transaction processor (TP). It supports both COBOL and C/C++ applications, as well as Ruby and Python.

Tuxedo is a transaction processing system or transaction-oriented middleware, comparable to IBM's MQ and designed for high availability and to provide scalable applications. It can perform a lot of transactions per second on commonly available distributed systems. Originally, it was developed and designed by AT&T telephone company to act as an online transaction processing (OLTP) system.

Tuxedo uses the technique of message routing and queuing these messages into its system. Message requests are sent to Tuxedo named services and then it uses memory-based communication to queue the requests to Tuxedo servers. The requester is unaware of where the server that actually processes the request is located or how it is implemented, so this is an asynchronous request. This is, in fact, the principle of how SOA is implemented in nowadays modern systems before even the concept of SOA was familiar to any one.

In 2008, Oracle acquired BEA and for a long time it seemed undetermined what Oracle would do with this Tuxedo systems. Although a quite unknown software product, large bank messaging systems such as the International Bank Transfers Systems (SWIFT) uses Tuxedo for message processing

Oracle Tuxedo 12c will be part of the new Exabus stack, a component embedded in the Exalogic solution Oracle launched earlier.

Tuxedo's migration abilities such as workbench tools for the migration process are integrated in Eclipse, so developers can migrate mainframe CICS resources and COBOL copybooks to the Oracle platform. In general, you can say that Tuxedo in Exalogic is meant to migrate non-Java applications (especially Mainframe applications).

Tuxedo in the Oracle version

The structure of Tuxedo consists of the following components:

- **Communication concentrators:** For remote clients (Java, CORBA, or Web Services).
- **Gateways:** To facilitate the sharing of services across domains, Tuxedo provides domain gateways.
- **Failure recovery:** Each machine monitors the state of all servers and can automatically restart failed servers.
- **Transaction monitoring and coordination:** Tuxedo applications can make use of the transactions (to databases or other subsystems) to be controlled by the application or automatically controlled by the Tuxedo configuration, that is, container controlled transactions.
- **Queuing subsystem:** Tuxedo uses a queuing subsystem called /Q. This facility provides transient and persistent queues. You can compare these with other messaging queuing products like JMS or MQ.

Oracle Tuxedo's new or enhanced features

- **Mainframe re-hosting:** To migrate the C/C++ and COBOL applications.
- **SALT:** For web services, SOAP/HTTP(S) gateway and for developing SCA based applications in C++, Python, PHP, and Ruby. Supports modules for Apache Web Server, Oracle HTTP Server, and Oracle iPlanet Web Server.
- **Tuxedo Mainframe Adapters (TMA):** This provides a set of processes that run on Tuxedo that communicate with a mainframe.
- **JCA Adapter:** This is a wrapper to WebLogic Tuxedo Connector (WTC) as part of WebLogic Server. WTC can only be used on WebLogic, but the JCA adapter allows deploying WTC capabilities on other Java Apps Servers that support JAVA EE JCA.

Message Bus API: Oracle Coherence

Oracle Coherence is a product that can be used for application clustering and reliable data sharing, caching state in the Application tier and relieve load on lower-tier systems such as databases, mainframes, Web Servers, or Web Services.

Coherence is used for a so called Data Grid. A **Data Grid** is a system composed of multiple servers that work together to manage information and related operation.

Coherence can be used for the following cases:

- **Caching:** Applications request data from the Data Grid rather than backend data sources
- **Analytics:** Applications ask the Data Grid questions from simple queries in advanced scenario modeling
- **Transactions:** Data Grid acts as a transactional System of Record, hosting data and business logic
- **Events:** Automated processing based on events

Integration with Coherence can be achieved by Custom integration through the Coherence API or through existing *Switch-On* out-of-the-box integrations such as:

- TopLink Grid in combination with JPA object-relational data. This simplifies Coherence use in shared database environment and propagates DB updates to Coherence. GoldenGate captures the changes to database tables and TopLink maps database changes to cached.
- WebLogic Server: In-memory HTTP session on Grid.
- Service Bus: Service Result Caching. Result caching is controlled at business services level and allows for fine-grained control for composite services. It caches the same set of services being called many times so it doesn't have to go over the line again, which gives better performance. A subset of results will make up final results of a cached service.

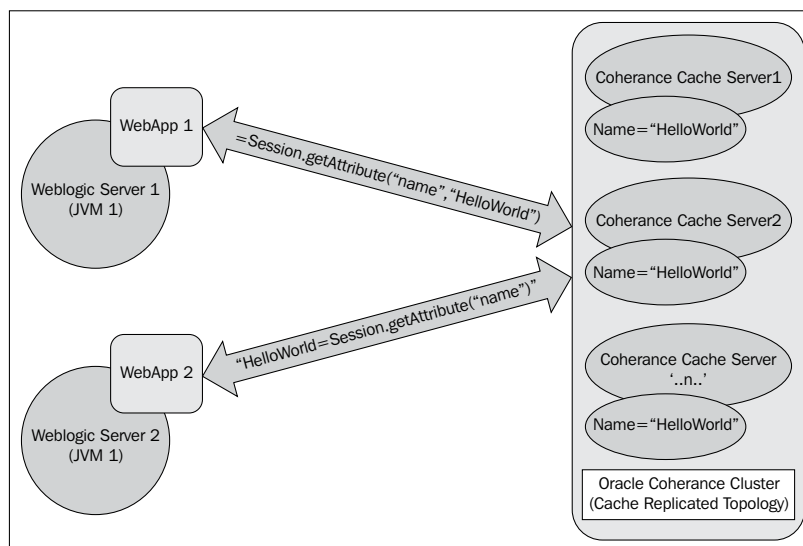
Coherence in Exalogic uses direct memory access and bypasses the kernel to improve speed and performance. The newest version of Coherence (3.7.*) has ActiveCache integration with WebLogic Server. It has Coherence cluster MBeans within WebLogic Server. The Node Manager is enhanced for starting/stopping Coherence cache servers from the Administration Console or doing it remotely.

Coherence supports Java APIs and Exalogic Elastic Cloud Software and has focused on Exalogic with the following features:

- Enable low-latency computing
- Optimized implementation for Exalogic Infiniband

- Scale individual nodes on multicore machines
- Advanced networking capabilities
- Messaging
- Direct memory access throughout RDMA
- Asynchronous APIs

Another feature is Coherence*Web integration and is used for HTTP session management dedicated to managing session state in clustered environments.



The application grid running Coherence*Web is shared between two clusters and active HTTP session are accessible from both clusters. Rolling traffic from primary cluster into a backup does not degrade performance.

SDP API: WebLogic

WebLogic 12c is enhanced with Exalogic features, all bundled in the SDP API.

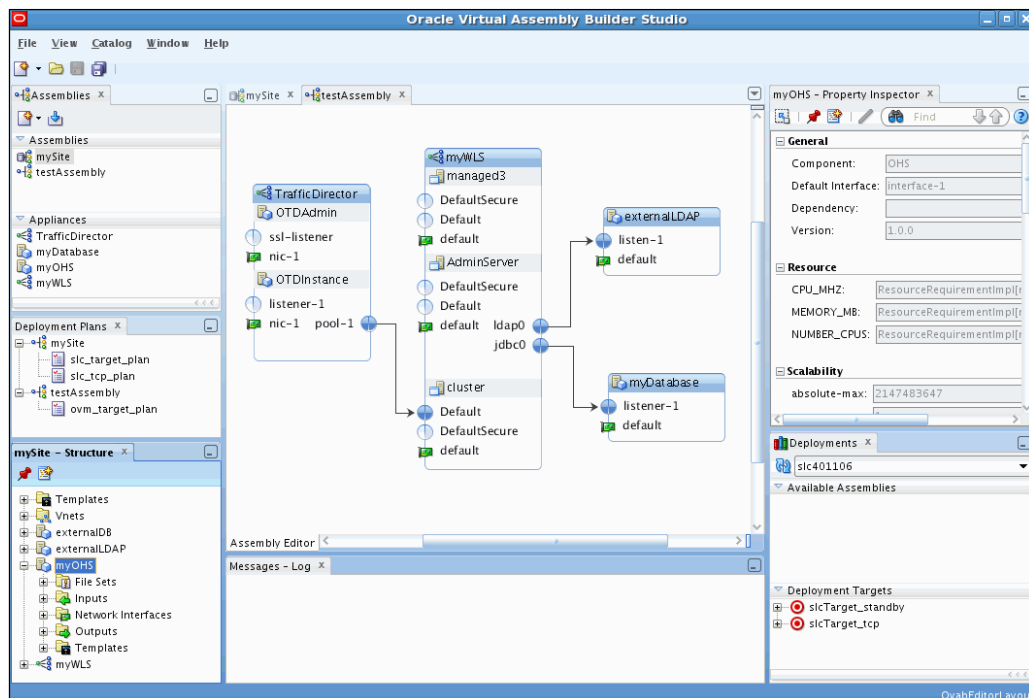
SDP or Sockets Direct Protocol is a low-level, remote-computing protocol, originally designed for Infiniband fabrics. It defines a standard wire protocol for streams. You can call it an RDMA-accelerated alternative to TCP/IP. It is part of the OpenFabrics Enterprise Distribution (OFED) and it has support in JDK 7 on Solaris and Linux.

WebLogic uses SDP for inter-process communications over the Infiniband interface. Parallel muxers give a faster message flow because it reduces lock contention, and larger packet sizes reduce processing messages onto the network.

- Increased WebLogic scalability, throughput, and responsiveness: Improvements to WebLogic's networking capabilities, request handling, and thread management mechanisms, which enable it to scale better on the high multicore compute nodes which are working on the InfiniBand fabric that ties all the compute nodes together. Each WebLogic Server can handle more client requests while at the same time also reducing the time taken to respond to each individual request.
- Increased performance on Session Replication mechanisms: WebLogic's session replication mechanism for Exalogic is enhanced to deal with the broad bandwidth of InfiniBand used for parallel connections over the network. Java web applications take great advantage of this enhancement because the states in a cluster can be replicated with the speed of light between the applications.

Oracle Virtual Assembly Builder

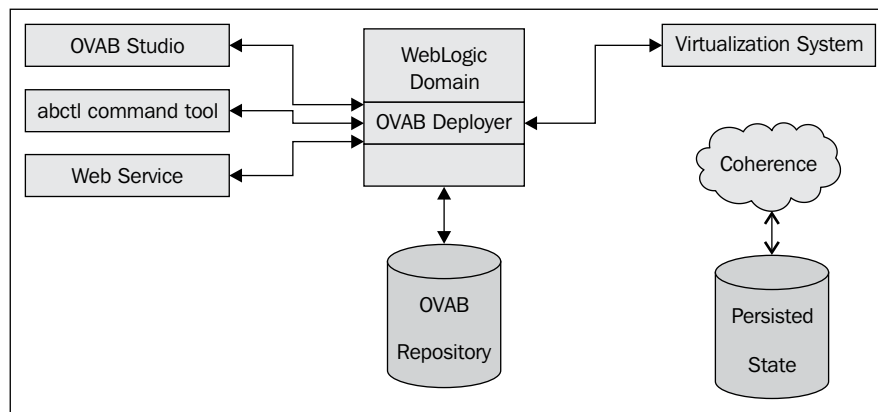
Oracle Virtual Assembly Builder is a tool that creates multitier Oracle Fusion Middleware environments as prepackaged Oracle VM templates. These packages are called assemblies. They contain preconfigured virtual machines and a complete prepared environment which can be deployed on Exalogic on demand. This can be done as many times as you wish.



Virtual Assembly builder works closely together with WebLogic. You can download it from Oracle Technet: **Oracle Technology Network | Middleware | Virtual Assembly Builder | Downloads.**

In fact, you will install the WebLogic Server software, install the latest OVAB software like 11g Release 1 (11.1.1.6) and create an OVAB Domain as you would normally do with any WebLogic Domain. After you have started your domain, you are ready to build and deploy any virtual appliance.

The following diagram shows the components that belong to a OVAB Architecture:



Oracle Traffic Director

Oracle Traffic Director is one of the newest products that was launched in 2011, to make the concept of an engineered system complete. Oracle Traffic Director is some sort of software load balancer and is the actual replacement of Oracle's previous load-balance product called Oracle WebCache.

Oracle Traffic Director will have to compete with traditional load balancers such as Cisco's BigIP or other hardware load balancer solutions.

In fact, traditional load balancers do their work from a network point of view, while OTD is a so called Application Routing Engine, which is completely right as it operates on network layer 7, the application layer.

OTD supports the latest SSL encryption. It uses 40 GB/s Infiniband connections to support not only external traffic, but lots of internal traffic, in particular SOA (such as service calls from and to BPEL and OSB). In a typical SOA messaging system with extreme load with millions of messages a day, transformed and routed by Oracle Service Bus and transferred to BPEL to be processed further, Exalogic with its extreme network capabilities could be a solid solution for your entire Enterprise.

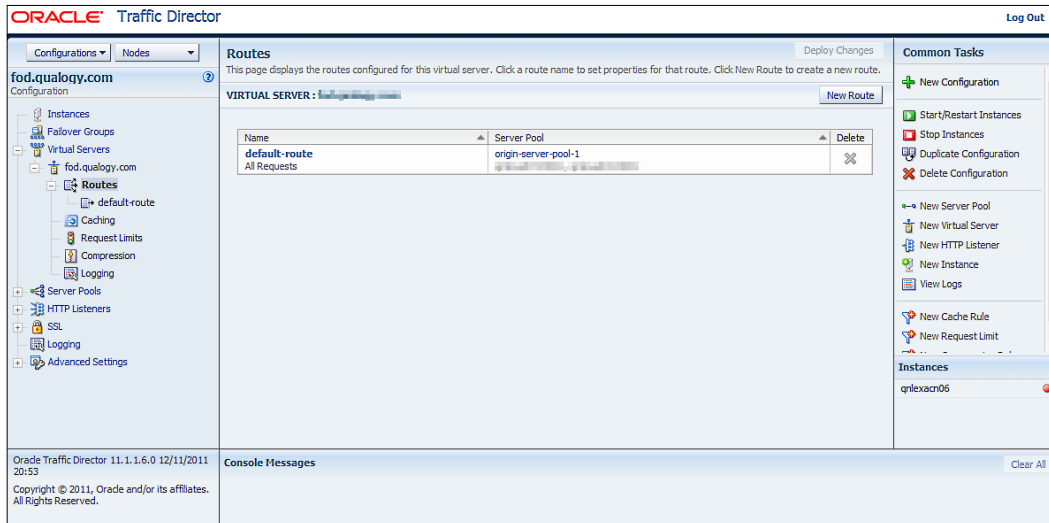
And finally, with Oracle Traffic Director you will get complete control of your environment, where a Middleware Administrator becomes suddenly a Network Administrator. It all comes together into one department, with no boundaries between the different departments anymore, the entire Fusion Middleware stack is tightly integrated from hardware to network.

Oracle Traffic Director is originally based on iPlanet, a mature product with a history spanning a large number of years and almost as many names, and something Oracle acquired as part of Sun. The engineering effort has been going into management tooling, such as scripting.

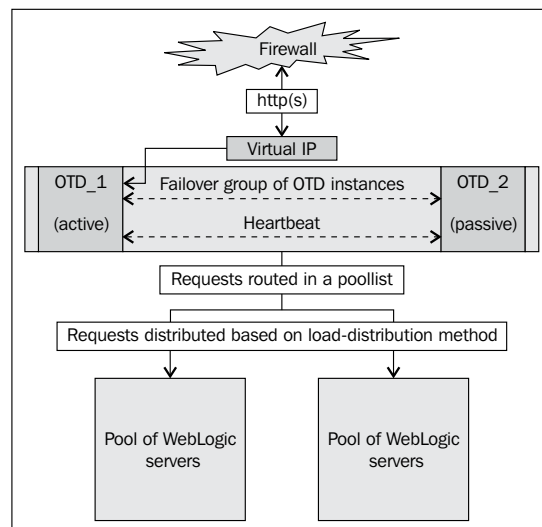
Some facts about Oracle Traffic Director are as follows:

- OTD supports SSL encryption
- 40 GB/s Infiniband connections instead of the 1 GB/s (but not to the outside world)
- A virtual appliance (VA) running on Oracle VM Server
- SSL-offloading, caching, and redirecting
- Tight integration with Oracle HTTP Server and other Apache versions
- Reverse-proxy routing/load-balancing
- Request rate limiting/throttling for protecting your resources from overload
- Cluster management
- Integration of Single Sign On and Access Management
- Infiniband/SDP support
- OVM/OVAB support for building virtual assemblies

The following is the screenshot of Oracle Traffic Director's homepage:



You can serve a pool of WebLogic servers by OTD, which acts just like a hardware load-balancer. You can see in the following diagram a typical OTD architecture:



Oracle WebLogic/Exalogic optimizations

From WebLogic Server version 10.3.4, Oracle has done a great job to optimize it for an Exalogic environment. The most important ones are discussed here.

Increased server scalability, throughput, and responsiveness

These improvements lie in the fact that various components, such as networking, request handling, memory, and thread management in WebLogic as well as in JRockit, are better scaled for the high-end Infiniband interface and the multiple CPU core nodes. WebLogic uses socket handlers that are Java NIO-based for more efficient request processing, multicore aware thread pools, and shared byte buffers to reduce data copies between subsystem layers.

- **Java NIO:** The new Java NIO (New Input Output) is meant for more efficient use of threads and greater throughput. The new I/O APIs are designed by the JSR-203 specifications and bundled in the `java.nio.file` package.
- **Self-tuning Thread Pool:** Another enhancement is a fully-optimized work scheduler, providing improvements to the Increment Advisor, which is used to manage the size of WebLogic Server's self-tuning Thread Pool. This aligns the Thread Pool to take advantage of the fast CPUs inside the Exalogic system. It can be set in the MBean `KernelMBeanmbeanKernelMBean.addWorkManagerThreadsByCpuCount`.
- **Data buffer copies:** The next good modification is the reduction of the number of data buffer copies that have been incorporated. WebLogic Server has changed to use byte buffers to collect data responses. These buffers are shared between the WebLogic's subsystem layers, in contrast to the old behavior where copies of data arrays were made and then passed between the subsystem layers.

To reduce network overhead, some sort of I/O pattern is used (Scatter/Gather I/O—read and write in multiple memory buffers) between WebLogic and the JVM when it sends data over the network.

These enhancements can be set by using:

- `Dweblogic.ScatteredReadsEnabled=true/false` or `KernelMBean.setScatteredReadsEnabled` for Scattered Reads
- `Dweblogic.GatheredWritesEnabled=true/false` or `KernelMBean.setGatheredWritesEnabled` for Gathered Writes

Server session replication performance

WebLogic can use the In-Memory HTTP Session Replication mechanism for maintaining application session states in a cluster. This mechanism is optimized for the large Infiniband bandwidth between managed servers in a cluster. The session data can be replicated in parallel to a second server, using parallel socket connections instead of just a single connection. WebLogic also avoids a lot of unnecessary processing that usually takes place on the server receiving session replicas, by using lazy deserialization. With the Infiniband optimized JRockit JVM, WebLogic does not use the TCP/IP stack, but the InfiniBand's SDP, to enable the session payloads to be sent over the network with lower latency.

The lazy deserialization can be set by using `Dweblogic.replication.enableLazyDeserialization=true/false` or `ClusterMBean.setSessionLazyDeserializationEnabled`.

Better Oracle RAC and Exadata integration

From version 10.3.4, Oracle introduced some major improvements for either the traditional Oracle RAC as Exadata database systems such as RAC active load balancing and improved failover mechanisms (any RAC not just ExaData).

Another new technology that was introduced was called Active GridLink. This provides optimized Oracle RAC database connectivity and intelligent load balancing across RAC nodes to detect faster if a connection should failover when a RAC node fails. All these new technologies delivered reduced response time in the communication between Exalogic and Exadata. This is because communication runs over the Infiniband's own native network protocol called SDP to interact between the different components.

The low latency benefit for request-response times for calls between WebLogic and the database will be reached if you use large result sets. Those are transferred from database to WebLogic, which will also lead to faster response of applications.

If you want to use the SDP protocol, you will have to add the following code to your startup options or enable it in the WLS Admin Console:

```
-Djava.net.preferIPv4Stack=true
```

Active GridLink is a new feature area for scaling WebLogic Server up to Oracle Database RAC. Before this feature, WebLogic already was the only Java EE 5 application server on the market with declarative integration with RAC using a feature called Multi Data Sources and these continue to be supported. However, Active GridLink for RAC takes this integration to a new level. There are five key features of GridLink:

- **Single data source for an entire RAC cluster:** When you have many RAC nodes in your environment, you still need one data source. With GridLink and ONS, you do not need to worry about using all the RAC nodes, because these components will handle the failover and load balance mechanisms.
- **RAC workload awareness:** By monitoring the workload of a typical RAC node, GridLink can decide to balance it across the other nodes which leads to even better performance and protects your database from being overloaded.
- **Transaction affinity:** Global transactions have an affinity context assigned to a specific RAC instance enabling significantly better performance.
- **Fast Connection Failover:** This works on the basis of events occurring in the JDBC connection pool, manual or automatic events, such as if one RAC node fails, an event is raised and it will be detected by the FCF mechanism, and immediately the connection will switch to any other available RAC node. Also, if a DBA shuts down an RAC node for maintenance, FCF detects it and will switch connections.
- **Single Client Access Name (SCAN):** Besides Active GridLink and FCF, WebLogic 12c supports a typical RAC feature called SCAN. This is configured at the TNS level with a SCAN Listener address acting like a client. In this case, WLS needs to be configured against just one SCAN address instead of all the addresses of the different RAC nodes.

Reduced Exalogic to Exadata response times

The connection between an Exalogic and an Exadata system is over the superfast Infiniband interface using the native Sockets Direct Protocol (SDP). JDBC will interact much faster with the database on the Exadata. Along with JRockit enhancements, a Java EE application will respond much faster to end clients, even if it has to process large JDBC result sets.

Summary

So, with this final chapter, it has been tried to give you a first look inside the new release of Oracle WebLogic 12c. A key fact to mention is that it has been made ready for the future cloud and to act as the key application server on Oracle's engineered system Exalogic. One other major improvement is the Java EE 6 implementation plus Java SE 7. In future releases, Java EE 7 will come in scope and a lot of improvements are to be done, but with this product, I think Oracle has a strong platform in the enterprise world.

Another point is it will be more and more integrated to one whole. At the time of writing this book, there is, for instance, still an Administration Console plus an Enterprise Manager Console, and two Java Virtual Machines (JRockit and HotSpot), but in future releases Oracle will bring all these components together in one.

This is just one of the many things that are planned to happen in a future release.

I hope this book has given you a broad overview of Oracle WebLogic Server 12c; although not all 200 new features could appear in this book, I believe the most important ones are discussed.

Have fun trying it and try to bring your platform up to this magnificent level!

Index

Symbols

@PostConstruct method 24
@PreDestroy method 24
@Startup annotation 24

A

Active GridLink
about 115, 116
and RAC, integrating 63, 64
Fast Connection Failover feature 116
features 116
RAC workload awareness feature 116
Single Client Access Name (SCAN) 116
Single data source for an entire RAC cluster
feature 116
Transaction affinity feature 116
ADF Web GUI. *See* **Oracle Enterprise Man-
ager Cloud Control**
Administration Console 58
AffEnabled attribute 68
analytics 108
application
deploying 37
Application-Scoped Drivers 67
AppXray 43
assemblies 110

B

beans.xml file 19
bean validation 19
bind task 77
BTM 98
Business Transaction Management. *See*
BTM

C

caching 108
Capacity Increment Attribute 65
CAT
and Classloading 51
CDI
about 18
for Java EE (JSR 299) 19, 20
CertPathValidator security provider 73
Classloading
and CAT 51
Classloading Analysis Tool. *See* **CAT**
Classloading filtering
features 52
cloud development
with WebLogic 12c 55
Cloud Technology 36
Coherence*Web integration feature, Oracle
Coherence 109
communication concentrators 107
Composite Application dashboard,
WebLogic Server 12c 90
compute nodes 103
com.sun.faces.spi.InjectionProvider inter-
face 21
configuration features 11
ConnectionAffinityCallback interface 68
Connection Labeling 68
context-root element 35
Contexts and Dependency Injection. *See*
CDI

D

database management system. *See* DBMS

data buffer copies 114

Data Grid 108

Data Source Profile Logging 66

DBMS 7

Debug Scopes 68

deployment descriptor

for GlassFish Server 54

deprecated APIs

about 33

EJB 2.x Entity Beans CMP 33

Java EE Application Deployment (JSR-88)
33

JAXR 33

JAX-RPC 33

development environment

configuring, tips 39

using, tips 39

development features 9, 10

distributed caching 13, 14

E

EAR application classpath 51

Eclipse

and Oracle Enterprise Pack for Eclipse
(12.1.1.0) 41-43

Maven for 49

EclipseLink MOXy. *See* JAXB

EJB 2.x Entity Beans CMP, deprecated APIs
33

EJB 3.1 18

EJB Lite

about 25

features 25

EJBs

about 22, 23

annotation support, deployment with 26,
27

cron-style declarative and programmatic
timers 23

EJB 3.1 annotation support 26

EJB Lite 25

embeddable EJB 25

global JNDI names 24

in war, admin console support 26

simplified WAR packaging 24

singleton beans, with concurrency control
23

startup/shutdown callbacks 24, 25

embeddable EJB 25

Enterprise Java Beans. *See* EJB 3.1

Enterprise Manager 12c 12, 13

Enterprise Manager 12c (12.1.0.1) Grid

Control . *See* Oracle Enterprise

Manager 12c (12.1.0.1) Grid Control

Enterprise Manager (EM) 80

event monitoring, WebLogic Server 12c 98

events 108

Exabus 104

Exalogic. *See* Oracle Exalogic

Exalogic Applications 36

Exalogic Cloud Software

about 105

APIs 105

Exalogic Cloud Software, APIs

about 105

Message bus API 105

RDMA API 105

Sockets Direct Protocol (SDP) API 105

Exalogic Cloud Software components. *See*

Oracle Exalogic Cloud Software
components

Exalogic Elastic Cloud 55

Exalogic features 9, 14, 15, 61

Execution Context ID (ECID) 97

F

failover features 11

failure recovery 107

FAN

about 64

enabling 64

Fast Application Notification. *See* FAN

FastSwap option 39

Fatal Error Codes 66

features

configuration features 11

development features 9, 10

Exalogic features 9, 14, 15

failover features 11

- HotSpot JVM features 9
- Java EE 6 features 8, 9
- performance features 11
- time management features 12
- tooling features 11

G

- gateways 107
- GlassFish Server
 - deployment descriptor for 54
- grid computing (g) 7

H

- heap analysis 99
- HotSpot JVM features 9

I

- IDEs
 - Maven for 49
- Infiniband I/O Fabric and 10 GbE 102, 103
- Integrated Cloud Stack Management 99
- IntelliJ IDEA
 - features 44
- International Bank Transfers Systems 106

J

- JACC 19, 71
- JASPIC 70
- java:app 59
- java:global 59
- java:module 58
- Java API for RESTful Web Services (JAX-RS 1.1) 18
- Java API for RESTful Web Services (JSR 311) 32
- Java API for XML-based RPC. *See* JAX-RPC
- Java API for XML-based Web Services. *See* JAX-WS
- Java Authentication Service Provider Interface for Containers. *See* JASPIC
- Java Authorization Contract for Containers 1.4. *See* JACC
- JavaBean class 20

- Java Connection Architecture 1.6. *See* JCA

- Java EE 6
 - goals 17
 - in Cloud Technology 36
 - WebLogic Web Services with 75

- Java EE 6 API

 - modifications 18, 19

- Java EE 6 features 8, 9

- Java EE 6, specifications

 - about 19
 - admin console support for EJBs, in WAR 26
 - bean validation 19
 - Bean Validation 1.0 (JSR 303) 28
 - Contexts and Dependency Injection (CDI) 18
 - Contexts and Dependency Injection for Java EE (JSR 299) 19, 20
 - deployment, with annotation support 26, 27
 - deprecated APIs 33
 - Enterprise Java Beans 3.1 22-25
 - Enterprise Java Beans (EJB) 3.1 18
 - Java API for RESTful Web Services (JAX-RS 1.1) 18
 - Java API, for RESTful Web Services (JSR 311) 32
 - Java Authorization Contract for Containers 1.4(JACC) 19
 - Java Connection Architecture 1.6 (JCA) 19
 - Java EE Connector Architecture 1.6 33
 - Java Persistence API (JPA) 2 18, 28, 30
 - Java Server Faces (JSF) 2 18-22
 - JAXB 2.2 19
 - Servlet 3 18
 - Servlet 3.0 31, 32

- Java EE application Classpath 51, 52

- Java EE Application Deployment (JSR-88), deprecated APIs 33

- Java IDE support 41

- Java NIO 114

- Java Persistence API. *See* JPA 2

- Java Secure Socket Extension. *See* JSSE

- Java Server Faces 2. *See* JSF2

- javax.resource.spi.ActivationSpec interface 20

- javax.security.jacc package 71

JAXB

- about 77
- tasks 77

JAXB 2.2 19

JAXB, tasks

- bind task 77
- marshal task 77
- unmarshal task 77

JAXR, deprecated APIs 33

JAX-RPC 75

JAX-RPC, deprecated APIs 33

JAX-RS 76

JAX-WS 75

JCA

- about 19, 33
- features 33

JCA Adapter 107

JDBC 58, 59

JDBC services

- about 63
- Active GridLink and RAC, integrating 63, 64

JDK 7 certification 58

Jersey Java API for RESTful Web Services.

See JAX-RS

JmsDestinationAvailabilityHelper API 70

JMS services 69, 70

JMX command-line tool

- starting 96

JPA 2

- about 18, 28
- features 28, 30

JSF 2

- about 18, 21
- features 21, 22

JSSE 72-74

JVM Diagnostics 91-93

L

lightweight development

- with WebLogic 12c 38

M

mainframe re-hosting 107

ManagedConnectionFactory bean 20

marshal task 77

Maven

- about 44
- and NetBeans 50
- and WebLogic 12c, integrating 44, 45
- for Eclipse/OEPE 49
- for several IDEs 49

Message Bus API

- about 105
- Oracle Coherence 108

MessageServerBean 20

metadata annotations 33

metric extensions, WebLogic Server 12c 89

Middleware Cloning Wizard

- Oracle Middleware Home Gold Image, cloning from 85
- WebLogic Domain Provisioning Profile, cloning from 85

Middleware Diagnostics Advisor 93, 94

Middleware Home

- cloning, from existing installation 84

MinCapacity Attribute 65

N

Negotiate Identity Assertion provider 75

NetBeans

- and Maven 50

NetBeans IDE 7.1 43

New Message-Driven Bean (MDB)

- activation 70

NodeManager 58

O

OEPE

- Maven for 49

OMR 80

OMS 80

online transaction processing (OLTP) system 106

OpenFabrics Enterprise Distribution (OFED) 109

Oracle Coherence

- about 108
- Coherence*Web integration feature 109
- Data Grid 108
- in Exalogic 108
- integration 108

- used, for analytics 108
- used, for caching 108
- used, for events 108
- used, for transactions 108
- Oracle Enterprise Manager 12c**
 - about 79, 80
 - and WebLogic Server 12c, integrating 81
 - architectural overview 81
 - configuration changes, detecting 85
 - configuration changes, discovering 85
 - configuration management, features 82
 - system design 80
 - system design, features 81
 - WebLogic Server 12c, cloning 82-85
 - WebLogic Server 12c, provisioning 82-85
- Oracle Enterprise Manager 12c (12.1.0.1)**
 - Grid Control 80**
- Oracle Enterprise Manager Cloud Control 80**
- Oracle Enterprise Manager (OEM) 80**
- Oracle Enterprise Pack for Eclipse (12.1.1.0)**
 - and Eclipse 41-43
- Oracle Exalogic**
 - about 102
 - components 102
 - Exabus 104
 - Exalogic Cloud Software 105
 - Exalogic Cloud Software, APIs 105
 - Oracle Coherence 108, 109
 - Oracle Exalogic Cloud Software, components 104
 - Oracle Traffic Director 111, 112
 - Oracle Tuxedo 106, 107
 - Oracle Virtual Assembly Builder 110, 111
 - Oracle WebLogic/Exalogic optimizations 114
 - WebLogic 109
- Oracle Exalogic Cloud Software components 104**
- Oracle Exalogic, components**
 - compute nodes 103
 - Infiniband I/O Fabric and 10 GbE 102, 103
 - integrated storage 102
- Oracle Exalogic optimizations. *See* Oracle WebLogic, optimizations**
- Oracle Management Repository. *See* OMR**
- Oracle Notification Service (ONS) 63**
- Oracle RAC 115**
- Oracle Traffic Director**
 - about 111, 112
 - facts 112
 - homepage, screenshot 113
- Oracle Tuxedo**
 - about 106
 - in Oracle version 107
- Oracle Tuxedo, in Oracle version**
 - communication concentrators 107
 - failure recovery 107
 - gateways 107
 - JCA Adapter 107
 - mainframe re-hosting 107
 - queuing subsystem 107
 - SALT 107
 - transaction monitoring and coordination 107
 - Tuxedo Mainframe Adapters (TMA) 107
- oracle.ucp.ConnectionLabelingCallback interface 68**
- oracle.ucp.jdbc.LabelableConnection interface 68**
- Oracle Virtual Assembly Builder**
 - about 110, 111
 - components 111
- Oracle WebCache 111**
- Oracle WebLogic, optimizations**
 - about 114
 - data buffer copies 114
 - Exalogic to Exadata response times, reduced 116
 - Java NIO 114
 - Oracle RAC and Exadata integration, improved 115, 116
 - responsiveness, increased 114
 - self-tuning thread pool 114
 - server scalability, increased 114
 - server session replication performance 115
 - throughput, increased 114
- Oracle WebLogic Server 12c**
 - about 101
 - cloning 82-85
 - provisioning 82-85

OTD. *See* Oracle Traffic Director
OVAB. *See* Oracle Virtual Assembly
Builder

P

Partitioned Distributed Topics 69
performance features 11
POM 45-47
project object model. *See* POM

Q

Quad Data Rate (QDR) 104

R

RAC
about 7
and Active GridLink, integrating 63, 64
RDMA 105
RDMA API
about 105
Oracle Tuxedo 106
Real Application Cluster. *See* RAC
Remote Direct Memory Access. *See* RDMA
request monitoring, WebLogic Server 12c
90, 91
ResourceAdapter bean 20
Resource Adapter security 59
RSA JSSE Provider 72
Runtime Connection Load-Balancing
(RCLB) 68

S

SALT 107
SCA applications
building, in WebLogic 42
SDP 109
SDP API
about 105
WebLogic 109
security services 70
self-tuning thread pool 114
servlet 3 18
servlets 3.0 31, 32
Session Affinity Policy 68

shared library 34
Simple and Protected Negotiate. *See*
SPNEGO
Single Client Access Name (SCAN) 116
Sockets Direct Protocol. *See* SDP
SPNEGO 74
SSL 59, 73, 74
SSL Implementation 72
SSLMBean 73
SSO
with Microsoft Clients 74
standalone clients 60
start() method 21
startup/shutdown callbacks 24, 25
SWIFT. *See* International Bank Transfers
Systems

T

Thin T3 Client 60
time management features 12
TLog store 62
tooling features 11
Traffic Director. *See* Oracle Traffic Director
transaction monitoring and coordination
107
transactions 108
Tuxedo. *See* Oracle Tuxedo
Tuxedo Mainframe Adapters (TMA) 107

U

unmarshal task 77
unrestricted Client ID policy 70

V

validate() method 21
Virtual Assembly Builder. *See* Oracle
Virtual Assembly Builder

W

WAR application classpath 51
WebCache. *See* Oracle WebCache
WebLogic 109
WebLogic 12c
about 34, 35, 55

- and Jersey JAX-RS RI Version 1.9 76
- and Maven, integrating 44, 45
- cloud development with 55
- development environment, tips 39
- development, features 9, 10
- distributed caching 13, 14
- Enterprise Manager 12c 12, 13
- Exalogic, features 14, 15
- failover 11
- features 8, 9
- Java EE 6, features 8, 9
- lightweight development 38
- modules 34, 35
- new configuration features 58
- performance 11
- shared libraries 34, 35
- TLog store, new feature 62
- traffic management 12
- upgrading to 57
- WebLogic Server installers 56, 57
- Web Services 60
- WebLogic 12c, configuration features**
 - administration console 58
 - Java Authentication SPI for containers (JASPIC) support 59
 - JDBC 58
 - JDK 7 certification 58
 - NodeManager 58
 - resource adapter security 59
 - security 59
 - SSL 59
 - standalone clients 60
 - weblogic.management.username and weblogic.management.password 60
- WebLogic 12c, JDBC**
 - features 65-69
- weblogic.appc class 26**
- WebLogic Comparison Template 86**
- WebLogic Domain**
 - cloning 82
 - cloning, from existing installation 83
- weblogic.jdbc.rac.DebugJDBCONS 69**
- weblogic.jdbc.rac.DebugJDBCRCAC 69**
- weblogic.jdbc.rac.DebugJDBCRCREPLAY 68**
- weblogic.jdbc.rac.DebugJDBCUCP 68**
- weblogic.jdbc.transaction.DebugJTJDBC 68**
- WebLogic Maven plugin**
 - features 48
- weblogic.security.jacc 71**
- WebLogic Server 101**
- WebLogic Server 12c**
 - application components, dependency 96
 - application components, performance 96
 - Composite Application dashboard 90
 - deployed applications, monitoring 95, 96
 - diagnostics 86-88
 - diagnostic, snapshots 94
 - event, monitoring 98
 - JVM Diagnostics 91-93
 - log viewer 97
 - metric extensions 89
 - Middleware Diagnostics Advisor 93, 94
 - monitoring 86
 - out-of-box metrics 88, 89
 - performance, monitoring 86-88
 - performance summaries, customizable 88
 - request monitoring 90, 91
- WebLogic Server components 38**
- WebLogic Server Management Pack Enterprise Edition 81**
- WebLogic server tooling**
 - using 40
- WebLogic the Management Service. *See* OMS**
- Web Services**
 - about 75
 - in 12c 76
 - WebLogic Web Services, with Java EE 6 75
- Weighted Distributed Destinations 69**
- wls1211_dev_supplemental.zip 57**
- wls1211_dev.zip 57**
- wls1211_generic.jar 57**
- wls1211_linux32.bin 57**
- wls1211_solaris32.bin 57**
- wls1211_win32.exe 57**
- WLS-CAT tool 52, 53, 54**
- wlx option 40**

Z

- ZFS 102**
- ZIP installer**
 - downloading 39



Thank you for buying Oracle WebLogic Server 12c: First Look

About Packt Publishing

Packt, pronounced 'packed', published its first book "Mastering phpMyAdmin for Effective MySQL Management" in April 2004 and subsequently continued to specialize in publishing highly focused books on specific technologies and solutions.

Our books and publications share the experiences of your fellow IT professionals in adapting and customizing today's systems, applications, and frameworks. Our solution based books give you the knowledge and power to customize the software and technologies you're using to get the job done. Packt books are more specific and less general than the IT books you have seen in the past. Our unique business model allows us to bring you more focused information, giving you more of what you need to know, and less of what you don't.

Packt is a modern, yet unique publishing company, which focuses on producing quality, cutting-edge books for communities of developers, administrators, and newbies alike. For more information, please visit our website: www.packtpub.com.

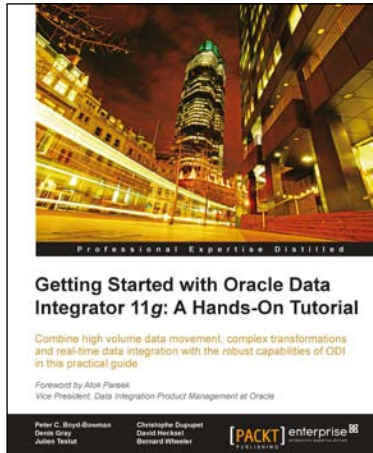
About Packt Enterprise

In 2010, Packt launched two new brands, Packt Enterprise and Packt Open Source, in order to continue its focus on specialization. This book is part of the Packt Enterprise brand, home to books published on enterprise software – software created by major vendors, including (but not limited to) IBM, Microsoft and Oracle, often for use in other corporations. Its titles will offer information relevant to a range of users of this software, including administrators, developers, architects, and end users.

Writing for Packt

We welcome all inquiries from people who are interested in authoring. Book proposals should be sent to author@packtpub.com. If your book idea is still at an early stage and you would like to discuss it first before writing a formal book proposal, contact us; one of our commissioning editors will get in touch with you.

We're not just looking for published authors; if you have strong technical skills but no writing experience, our experienced editors can help you develop a writing career, or simply get some additional reward for your expertise.



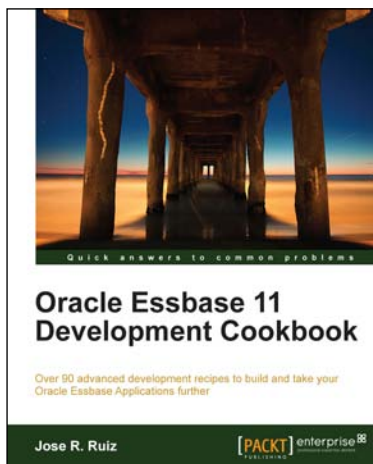
Getting Started with Oracle Data Integrator 11g: A Hands-On Tutorial

ISBN: 978-1-84968-068-4

Paperback: 384 pages

Combine high volume data movement, complex transformations and real-time data integration with the robust capabilities of ODI in this practical guide

1. Discover the comprehensive and sophisticated orchestration of data integration tasks made possible with ODI, including monitoring and error-management
2. Get to grips with the product architecture and building data integration processes with technologies including Oracle, Microsoft SQL Server and XML files
3. A comprehensive tutorial packed with tips, images and best practices



Oracle Essbase 11 Development Cookbook

ISBN: 978-1-84968-326-5

Paperback: 400 pages

Over 90 advanced development recipes to build and take your Oracle Essbase Applications further

1. This book and e-book will provide you with the tools needed to successfully build and deploy your Essbase application.
2. Includes the major components that need to be considered when designing an Essbase application.
3. This book can be used to build calculations, design process automation, add security, integrate data, and report off an Essbase cube.

Please check www.PacktPub.com for information on our titles

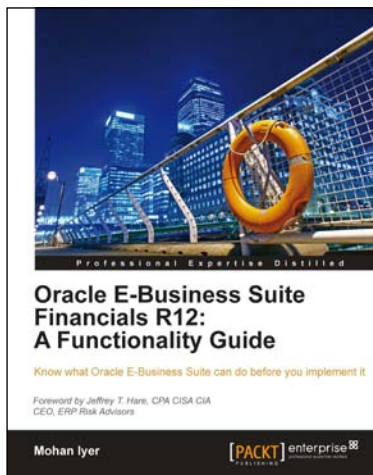


Oracle Application Integration Architecture (AIA) Foundation Pack 11gR1: Essentials

ISBN: 978-1-84968-480-4 Paperback: 274 pages

Develop and Deploy your Enterprise Integration Solutions using Oracle AIA

1. Full of illustrations, diagrams, and tips with clear step-by-step instructions and real time examples to develop full-fledged integration processes.
2. Each chapter drives the reader right from architecture to implementation.
3. Understand the important concept of Enterprise Business Objects that play a crucial role in AIA installation and models.



Oracle E-Business Suite Financials R12: A Functionality Guide

ISBN: 978-1-84968-062-2 Paperback: 336 pages

Know what Oracle E-Business Suite can do before you implement it

1. Take a deep dive into the key elements of Oracle EBS financial transaction processing
2. Understand the functionality and critical configuration steps
3. Master Oracle EBS product highlights and their effective usage

Please check www.PacktPub.com for information on our titles